

---

# Efficient Pairwise Multilabel Classification

---

Dissertation

Eneldo Loza Mencía

Knowledge Engineering Group  
Technische Universität Darmstadt  
eneldo@ke.tu-darmstadt.de



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



Knowledge  
Engineering

---

Vom Fachbereich Informatik der  
Technischen Universität Darmstadt  
zur Erlangung des akademischen Grades eines  
Doktors der Naturwissenschaften (Dr. rer. nat.)  
genehmigte

**Dissertation**

von

Diplom-Informatiker  
**Eneldo Loza Mencía**  
aus Frankfurt am Main

Referent: Prof. Dr. Johannes Fürnkranz  
Korreferent: Prof. Dr. Eyke Hüllermeier  
(Philipps-Universität Marburg)

Tag der Einreichung: 11. Juni 2012  
Tag der Verteidigung: 24. Juli 2012

Darmstadt 2013  
D17

---

## **Efficient Pairwise Multilabel Classification**

Genehmigte Dissertation von Eneldo Loza Mencía aus Frankfurt am Main

1. Referent: Prof. Dr. Johannes Fürnkranz

2. Referent: Prof. Dr. Eyke Hüllermeier

Weitere Prüfer:

Prof. Dr. Michael Goesele (Vorsitzender der Prüfungskommission)

Prof. Dr. Chris Biemann

Prof. Dr. Thorsten Strufe

Tag der Einreichung: 11. Juni 2012

Tag der Verteidigung: 24. Juli 2012

Darmstadt, 2013 – D17

Bitte zitieren Sie dieses Dokument als

Please cite this document as

URN: urn:nbn:de:tuda-tuprints-32260

URL: <http://tuprints.ulb.tu-darmstadt.de/3226>

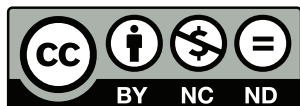
Dieses Dokument wird bereitgestellt von

This document is provided by

tuprints, E-Publishing-Service of the TU Darmstadt

<http://tuprints.ulb.tu-darmstadt.de>

[tuprints@ulb.tu-darmstadt.de](mailto:tuprints@ulb.tu-darmstadt.de)



Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

This publication is licensed under the following Creative Commons license:

Namensnennung – Nicht kommerziell – Keine Bearbeitung 3.0 Deutschland

Attribution – Non-commercial – No Derivative Works 3.0 Germany

(CC BY-NC-ND 3.0 DE)

<http://creativecommons.org/licenses/by-nc-nd/3.0/de/>

---

# Kurzfassung

Multilabel-Klassifizierung bezeichnet die Aufgabe, eine Zuordnung von Objekten zu Mengen von möglicherweise sich überlappenden Klassen zu lernen. Dieses Feld hat in letzter Zeit stark an Aufmerksamkeit gewonnen. Ein praktisches Anwendungsszenario wäre die Zuweisung von Schlüsselwörtern zu Dokumenten. Ein Problem, das häufig im Gebiet der Text-Klassifizierung anzutreffen ist. Durch die aufkommenden Web 2.0 Technologien wird dieses Gebiet um eine Reihe von Szenarien erweitert, welche sich hauptsächlich mit dem Empfehlen von Tags und Schlagwörtern konzentrieren. Der Trend geht dabei unausweichlich zu noch mehr Datenpunkten und noch mehr Labels. Die vorliegende Arbeit bietet eine umfassende Einleitung in das Thema Multilabel-Klassifizierung mit einer detaillierten Formalisierung und einer ausführlichen Erläuterung der aktuellen Verfahren, die den Stand der Technik repräsentieren.

Ein gängiges Verfahren, um Multilabel-Probleme zu lösen, stellt die Zerlegung des Originalproblems in mehrere Teilprobleme dar. Diese Teilaufgaben sind üblicherweise leicht mit konventionellen Techniken zu lösen. Im Vergleich zu der direkten Methode, dem Lernen eines Klassifizierers pro Klasse, der dann für jede Klasse unabhängig von den anderen die Relevanz vorhersagt (Binary Relevance), legt diese Arbeit ihren Schwerpunkt auf den Ansatz der paarweisen Zerlegung. Hierbei wird eine Entscheidungsfunktion für jedes Paar von Klassen gelernt. Der Hauptvorteil dieser Methode, die Verbesserung der Qualität der Vorhersagen, steht allerdings im Gegensatz zum Hauptnachteil, nämlich der quadratischen Anzahl von Klassifizierern. Diese Anzahl berechnet sich in Abhängigkeit der Anzahl der Labels. Diese Dissertation stellt ein Framework von effizienten und skalierbaren Lösungen für die Verarbeitung von hunderten und sogar tausenden von Labels vor, die trotz der quadratischen Abhängigkeit verarbeitet werden können.

Wie sich herausstellt, kann das Trainieren eines paarweisen Ensembles von Klassifizierern in linearer Zeit geschehen. Der Unterschied zum simplen Binary Relevance (BR) Verfahren beträgt hierbei nur einen kleinen Faktor, der der durchschnittlichen Zahl von assoziierten Labels pro Objekt entspricht. Zusätzlich konnte durch die Anwendung eines intelligenten und dynamischen Auswertungsschemas, inspiriert durch das System der Sport-Ligen, die quadratische Anzahl an Auswertungen von Basis-Klassifizierern auf eine in der Praxis log-lineare Abhängigkeit reduziert werden. Die Kombination mit einem einfachen aber schnellen und mächtigen linearen Klassifizierer erlaubte die Echtzeit-Verarbeitung von Daten mit sehr vielen, hoch-dimensionalen Datenpunkten. Eine Aufgabenstellung, die davor dem paarweisen Lernen nicht zugänglich war.

Der verbleibende Flaschenhals, die explodierenden Speicheranforderungen, wurde durch die Ausnutzung einer interessanten Eigenschaft von linearen Klassifizierern überwunden, nämlich die Möglichkeit der dualen Reformulierung als Linearkombination der Trainingsbeispiele. Die Tauglichkeit dieses Verfahrens wurde auf dem neuartigen

---

Textdatensatz *EUR-Lex* demonstriert, welches insbesondere die Skalierbarkeit des paarweisen Ansatzes auf die Probe stellt. Mit seinen fast 4.000 Labels und 20.000 Dokumenten stellt *EUR-Lex* eine der anspruchsvollsten Testdatensätze für Multilabel-Lernen dar. Die duale Formulierung ermöglicht es, ein Modell im Speicher zu halten, welches den 8 Millionen Basislernern, die bei der konventionellen Lösung für *EUR-Lex* nötig wären, entspricht, und das bei gleichen Speicherbedarf wie Binary Relevance. Darüber hinaus wurde BR in den Experimenten klar geschlagen. Ein weiterer Beitrag dieser Arbeit, basierend auf einer hierarchischen Zerlegung und Anordnung der Originalaufgabenstellung, ermöglicht sogar die Reduzierung der Abhängigkeit von der Anzahl der kleiner als linear. Dieser Ansatz öffnet einer großen Auswahl an neuen Herausforderungen und Anwendungen die Türen, aber es werden dabei auch die Vorteile des paarweisen Lernens beibehalten, nämlich die exzellente Qualität der Vorhersagen. Im Vergleich mit dem konventionellen, flachen Ansatz konnte sogar gezeigt werden, daß sich bei Problemen mit vielen Labels ein besonders positiver Effekt beim Ausbalancieren von Recall und Precision einstellt.

Die verbesserte Skalierbarkeit und Effizienz ermöglichte es, den paarweisen Ansatz auf eine Menge von großen Multilabel-Problemen anzuwenden, die alle eine gemeinsame, parallele Datenbasis aber unterschiedliche Domänen von Labels besitzen. Dieses Szenario von parallelen Tasks stellt einen ersten Schritt dar, die Fähigkeiten des paarweisen Ansatzes für die Ausnutzung von Label-Abhängigkeiten zu untersuchen, mit ersten vielversprechenden Ergebnissen. Die Verwendung von Multilabel-Verfahren für die automatische Annotation von Texten stellt eine weitere offensichtliche, aber bislang verkannte Verbindung zu Multi-Task und Multi-Target-Learning dar. In der vorgeschlagenen Lösung wird die gleichzeitige Markierung von Wörtern mit unterschiedlichen aber möglicherweise überlappenden Annotationsschemata als Multilabel-Problem betrachtet. Dieser Ansatz wird voraussichtlich von Verfahren, die Label-Abhängigkeiten berücksichtigen, profitieren können. Die Fähigkeit des paarweisen Ansatzes hierfür ist klarerweise auf paarweise Relationen beschränkt. Deshalb wird in dieser Arbeit eine Technik untersucht, die Konstellationen von Labels erforscht, die nur lokal in Untergruppen der Datenpunkte zu finden sind. Zusätzlich zu dem festgestellten positiven Effekt dieser zusätzlichen Informationen bietet die experimentelle Auswertung auch interessante Erkenntnisse über das unterschiedliche Verhalten von aktuellen Verfahren bezüglich der Optimierung und besonderen Bevorzugung von Multilabel-Evaluationsmaßen, ein kontroverses Thema im Gebiet der Multilabel-Klassifizierung.

---

# Abstract

Multilabel classification learning is the task of learning a mapping between objects and sets of possibly overlapping classes and has gained increasing attention in recent times. A prototypical application scenario for multilabel classification is the assignment of a set of keywords to a document, a frequently encountered problem in the text classification domain. With upcoming Web 2.0 technologies, this domain is extended by a wide range of tag suggestion tasks and the trend definitely is moving towards more data points and more labels. This work provides an extended introduction into the topic of multilabel classification, a detailed formalization and a comprehensive overview of the present state-of-the-art approaches.

A commonly used solution for solving multilabel tasks is to decompose the original problem into several subproblems. These subtasks are usually easy to solve with conventional techniques. In contrast to the straightforward approach of training one classifier for independently predicting the relevance of each class (binary relevance), this work focuses particularly on the pairwise decomposition of the original problem in which a decision function is learned for each possible pair of classes. The main advantage of this approach, the improvement of the predictive quality, comes at the cost of its main disadvantage, the quadratic number of classifiers needed (with respect to the number of labels). This thesis presents a framework of efficient and scalable solutions for handling hundreds or thousands of labels despite the quadratic dependency.

As it turns out, training such a pairwise ensemble of classifiers can be accomplished in linear time and only differs from the straightforward binary relevance approach (BR) by a factor relative to the average number of labels associated to an object, which is usually small. Furthermore, the integration of a smart scheduling technique inspired from sports tournaments safely reduces the quadratic number of base classifier evaluations to log-linear in practice. Combined with a simple yet fast and powerful learning algorithm for linear classifiers, data with a huge number of high dimensional points, which was not amenable to pairwise learning before, can be processed even under real-time conditions.

The remaining bottleneck, the exploding memory requirements, is coped by taking advantage of an interesting property of linear classifiers, namely the possibility of dual reformulation as a linear combination of the training examples. The suitability is demonstrated on the novel *EUR-Lex* text collection, which particularly puts the main scalability issue of pairwise learning to test. With its almost 4,000 labels and 20,000 documents it is one of the most challenging test beds in multilabel learning to date. The dual formulation allows to maintain the mathematical equivalent to 8 million base learners needed for conventionally solving *EUR-Lex* in almost the same amount of space as binary relevance. Moreover, BR was clearly beaten in the experiments.

---

A further contribution based on hierarchical decomposition and arrangement of the original problem allows to reduce the dependency on the number of labels to even sub-linearity. This approach opens the door to a wide range of new challenges and applications but simultaneously maintains the advantages of pairwise learning, namely the excellent predictive quality. It was even shown in comparison to the flat variant that it has a particularly positive effect on balancing recall and precision on datasets with a large number of labels.

The improved scalability and efficiency allowed to apply pairwise classification to a set of large multilabel problems with a parallel base of data points but different domains of labels. A first attempt was made in this parallel tasks setting in order to investigate the exploitation of label dependencies by pairwise learning, with first encouraging results. The usage of multilabel learning techniques for the automatic annotation of texts constitutes a further obvious but so far missing connection to multi-task and multi-target learning. The presented solution considers the simultaneous tagging of words with different but possibly overlapping annotation schemes as a multilabel problem. This solution is expected to particularly benefit from approaches which exploit label dependencies. The ability of pairwise learning for this purpose is obviously restricted to pairwise relations, therefore a technique is investigated which explores label constellations that only exist locally for a subgroup of data points. In addition to the positive effect of the supplemental information, the experimental evaluation demonstrates an interesting insight with regards to the different behavior of several state-of-the-art approaches with respect to the optimization of particular multilabel measures, a controversial topic in multilabel classification.

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Challenges in Pairwise Multilabel Classification . . . . .	3
1.2	Contributions and Organization of the Work . . . . .	6
<b>2</b>	<b>Fundamentals of Multilabel Classification</b>	<b>11</b>
2.1	Input Object Space . . . . .	11
2.2	Classification Mapping . . . . .	11
2.3	Learning a Model and Predicting . . . . .	12
2.4	Output Label Space . . . . .	14
2.5	Types of Classification Problems . . . . .	15
2.5.1	Binary Classification . . . . .	15
2.5.2	Multiclass Classification . . . . .	17
2.5.3	Multilabel Classification . . . . .	17
2.5.4	Label Ranking . . . . .	19
2.5.5	Multilabel Label Ranking . . . . .	21
2.5.6	Hierarchical Classification . . . . .	22
2.6	Label Dependencies . . . . .	23
2.7	Evaluation Measures of Predictive Quality . . . . .	25
2.7.1	Aggregation and Averaging . . . . .	25
2.7.2	Cross Validation . . . . .	27
2.7.3	Bipartition Evaluation Measures . . . . .	27
2.7.4	Ranking Quality Measures . . . . .	31
2.7.5	Discussion . . . . .	35
2.8	Multilabel Learning Algorithms . . . . .	36
2.8.1	Transformational Approaches . . . . .	37
2.8.2	Holistic Approaches . . . . .	37
2.8.3	Generative Approaches . . . . .	38
2.8.4	Ensembles . . . . .	39
2.8.5	Instance Input Space Transformations . . . . .	40
2.8.6	Label Output Space Transformations . . . . .	40
2.8.7	Alternative Structures and Formulations . . . . .	41
2.9	Datasets and Application Scenarios . . . . .	42
2.9.1	Benchmark Datasets . . . . .	43
2.9.1.1	Text . . . . .	45
2.9.1.2	Multimedia . . . . .	46
2.9.1.3	Biology . . . . .	47

2.9.2	Sources and Repositories . . . . .	47
2.10	Statistical Comparison of Classifiers . . . . .	47
2.10.1	Sign Test . . . . .	48
2.10.2	Wilcoxon Signed-Ranks Test . . . . .	49
2.10.3	Friedman and Post-Hoc Tests . . . . .	50
<b>3</b>	<b>Decompositive Approaches to Multilabel Classification</b>	<b>53</b>
3.1	Binary Relevance Decomposition . . . . .	54
3.1.1	Computational Complexity . . . . .	56
3.2	Label Powerset Transformation . . . . .	57
3.2.1	Computational Complexity . . . . .	58
3.3	Error Correcting Output Codes . . . . .	58
3.4	Pairwise Multilabel Decomposition . . . . .	59
3.4.1	Decomposition . . . . .	59
3.4.2	Pairwise Preference Learning . . . . .	60
3.4.3	Aggregation . . . . .	61
3.4.4	Bipartitioning . . . . .	63
3.4.5	Calibration . . . . .	63
3.4.6	Computational Complexity . . . . .	65
3.4.6.1	Training . . . . .	66
3.4.6.2	Predicting . . . . .	67
3.4.6.3	Super-linear Base Learner . . . . .	68
3.5	Discussion . . . . .	70
3.5.1	Easier Subproblems . . . . .	70
3.5.2	Non-Competent Base Classifiers . . . . .	71
3.5.3	Instances in the Label Intersections . . . . .	71
3.5.4	Bipartitioning and Calibration . . . . .	73
3.5.5	Label Ranking . . . . .	74
3.5.6	Aggregation and Voting . . . . .	77
3.5.7	Comparison to Binary Relevance Decomposition . . . . .	78
3.5.8	Comparison to Ternary ECOC . . . . .	80
3.5.9	Summary: Advantages and Disadvantages . . . . .	81
<b>4</b>	<b>Pairwise Learning of Efficient Perceptrons</b>	<b>83</b>
4.1	Perceptrons . . . . .	84
4.1.1	Linear Classifiers . . . . .	85
4.1.2	Perceptron Training Rule . . . . .	85
4.1.3	Bias and Threshold . . . . .	86
4.1.4	Dual Form . . . . .	86
4.1.5	Maximum Margin Hyperplane and Problem Hardness . . . . .	87
4.1.6	Support Vector Machines . . . . .	87
4.1.7	Comparison . . . . .	88
4.2	Binary Relevance Ranking . . . . .	89



4.3	Multiclass Multilabel Perceptrons . . . . .	90
4.4	Multilabel Pairwise Perceptrons . . . . .	91
4.5	Comparison . . . . .	92
4.5.1	Discussion . . . . .	92
4.5.2	Computational Complexity . . . . .	93
4.5.2.1	Memory Requirements . . . . .	94
4.5.2.2	Training . . . . .	95
4.5.2.3	Prediction . . . . .	95
4.5.2.4	Sparsity of Feature Vectors . . . . .	96
4.6	Evaluation . . . . .	96
4.6.1	Experimental Setup . . . . .	96
4.6.2	Ranking Quality . . . . .	98
4.6.3	Calibration Performance . . . . .	98
4.6.4	Computational Costs . . . . .	100
4.6.5	Learning Curve . . . . .	101
4.6.6	Overfitting Analysis . . . . .	102
4.6.7	Concept Drift Analysis . . . . .	102
4.6.8	Results on other Datasets . . . . .	104
4.6.9	Discussion on Cardinality Prediction . . . . .	107
4.7	Summary . . . . .	108
<b>5</b>	<b>Efficient Aggregation Strategies</b>	<b>111</b>
5.1	Quick Weighted Voting . . . . .	112
5.1.1	QVoting for Multiclass Classification . . . . .	113
5.1.2	QVoting for Multilabel Classification . . . . .	114
5.1.3	Extensions . . . . .	115
5.2	Computational Complexity . . . . .	116
5.3	Experimental Setup . . . . .	117
5.3.1	Datasets . . . . .	117
5.3.2	Algorithmic Setup . . . . .	117
5.4	Evaluation . . . . .	118
5.4.1	Computational Efficiency . . . . .	120
5.4.2	Predictive Quality . . . . .	123
5.4.3	Support Vector Machines . . . . .	124
5.5	Discussion and Related Work . . . . .	126
5.6	Summary . . . . .	127
<b>6</b>	<b>Highly Scalable Dual Models</b>	<b>129</b>
6.1	The EUR-Lex Repository . . . . .	130
6.1.1	Retrieval . . . . .	132
6.1.2	Statistics . . . . .	132
6.1.3	Preprocessing . . . . .	136

6.2	Dual Multilabel Pairwise Perceptrons . . . . .	136
6.2.1	Calibration . . . . .	138
6.2.2	Discussion and Further Extensions . . . . .	139
6.3	Computational Complexity . . . . .	140
6.3.1	Memory Requirements . . . . .	141
6.3.2	Training . . . . .	142
6.3.3	Predicting . . . . .	142
6.4	Evaluation . . . . .	143
6.4.1	Experimental Setup . . . . .	143
6.4.2	Ranking Quality . . . . .	143
6.4.3	Bipartition Prediction Quality . . . . .	145
6.4.4	Computational Costs . . . . .	146
6.4.4.1	Training and Prediction Costs . . . . .	147
6.4.4.2	Memory Costs . . . . .	149
6.5	Related Work . . . . .	150
6.6	Summary . . . . .	151
<b>7</b>	<b>Hierarchical Model Efficiency and Scalability</b>	<b>153</b>
7.1	HOMER: Hierarchy of Multilabel Classifiers . . . . .	154
7.1.1	Training . . . . .	154
7.1.2	Predicting . . . . .	155
7.1.3	Hierarchy Construction . . . . .	155
7.2	Computational Complexity . . . . .	156
7.2.1	Memory . . . . .	157
7.2.2	Training . . . . .	157
7.2.3	Predicting . . . . .	159
7.3	Evaluation . . . . .	159
7.3.1	Setup . . . . .	160
7.3.2	Results of HOMER with QCLR . . . . .	160
7.3.2.1	Training Time . . . . .	161
7.3.2.2	Testing Time . . . . .	161
7.3.2.3	Predictive Quality . . . . .	163
7.3.3	Comparison of HOMER against its Base Classifiers . . . . .	165
7.3.3.1	Predictive Quality . . . . .	165
7.3.3.2	Computational Time . . . . .	167
7.4	Related Work and Discussion . . . . .	168
7.5	Summary . . . . .	169
<b>8</b>	<b>Exploitation of Label Dependencies in Parallel Tasks</b>	<b>171</b>
8.1	Related Work . . . . .	172
8.2	Preliminaries . . . . .	173
8.3	Parallel Task Learning . . . . .	173
8.3.1	Pairwise Classification for Parallel Tasks . . . . .	174

8.3.2	Calibration . . . . .	175
8.3.3	HOMER with QCLR . . . . .	175
8.4	Datasets . . . . .	177
8.5	Evaluation . . . . .	178
8.5.1	Additional Experiments . . . . .	181
8.6	Summary . . . . .	181
<b>9</b>	<b>High-Order Dependencies with Patterns from Subgroup Discovery</b>	<b>183</b>
9.1	Preliminaries . . . . .	184
9.1.1	The LeGo Framework . . . . .	184
9.1.2	Multilabel Classification . . . . .	185
9.1.3	Problem Statement . . . . .	187
9.2	Local Pattern Discovery Phase . . . . .	187
9.2.1	Exceptional Model Mining . . . . .	187
9.2.2	Exceptional Model Mining meets Bayesian Networks . . . . .	188
9.3	Pattern Subset Discovery Phase . . . . .	190
9.4	Global Modeling Phase . . . . .	191
9.5	Experimental Setup . . . . .	192
9.5.1	Datasets and Learners . . . . .	192
9.5.2	Obtaining Raw Results . . . . .	192
9.6	Evaluation . . . . .	193
9.6.1	Feature Selection Methods . . . . .	194
9.6.2	Evaluation of the LeGo Approach . . . . .	194
9.6.3	Evaluation of the Decompositive Approaches . . . . .	197
9.6.4	Efficiency . . . . .	198
9.7	Discussion and Related Work . . . . .	199
9.8	Summary . . . . .	200
<b>10</b>	<b>Information Extraction and Syntactic Parsing</b>	<b>203</b>
10.1	Information Extraction . . . . .	204
10.1.1	Boundary Classification . . . . .	204
10.1.2	Feature Generation . . . . .	205
10.2	Multilabel Classification for Information Extraction . . . . .	205
10.3	Evaluation . . . . .	206
10.4	Related Work . . . . .	207
10.5	Summary . . . . .	208
<b>11</b>	<b>Summary and Conclusions</b>	<b>209</b>
11.1	Challenges Revisited . . . . .	209
11.2	Perspectives . . . . .	211
11.3	Conclusions . . . . .	212
	<b>Bibliography</b>	<b>213</b>



<b>Own Publications</b>	<b>241</b>
<b>Acknowledgments</b>	<b>245</b>
<b>Curriculum Vitae</b>	<b>247</b>
<b>Statement of Authorship</b>	<b>247</b>

---

## List of Figures

1.1	Schematic complexity diagram for multiclass classification . . . . .	5
2.1	Taxonomy of problem settings . . . . .	18
2.2	Examples of multipartite label ranking . . . . .	21
2.3	Diagrams of predicted label rankings and measures . . . . .	31
3.1	Subproblems in binary relevance multilabel classification . . . . .	55
3.2	Subproblems in pairwise multilabel classification . . . . .	59
3.3	Pairwise multilabel training . . . . .	61
3.4	Pairwise voting table . . . . .	63
3.5	Calibrated pairwise multilabel training . . . . .	64
3.6	Pairwise voting table with calibrated label . . . . .	64
4.1	Subproblems in the multiclass multilabel perceptrons approach . . . . .	89
4.2	Pseudocode of the MMP algorithm . . . . .	90
4.3	Pseudocode of the training method of the MLPP algorithm . . . . .	91
4.4	MLPP ensemble as neural network . . . . .	93
4.5	Predicted and actual number of classes on <i>rcv1</i> . . . . .	99
4.6	Recall/Precision and ROC curves on <i>rcv1</i> . . . . .	100
4.7	Learning curve for the full dataset . . . . .	102
4.8	Error on training and test data depending on the number of epochs . . . . .	102
4.9	Comparison of random decision trees and CMLPP on drifted <i>rcv1</i> . . . . .	105
4.10	Learning curve of CMLPP on drifted <i>rcv1</i> . . . . .	105
5.1	Pseudocode of the QVoting algorithm . . . . .	113
5.2	Pseudocode of the QCLR2 aggregation algorithm . . . . .	114
5.3	Prediction complexity of QVoting and QCMLPP . . . . .	121
5.4	Prediction complexity of QCMLPP . . . . .	122
6.1	Excerpt of a <i>EUR-Lex</i> sample document . . . . .	131
6.2	Visualization of the EUROVOC graph. . . . .	133
6.3	Distribution of the labelset sizes for the three <i>EUR-Lex</i> datasets . . . . .	134
6.4	Diagram of the sizes of the labels on <i>EUR-Lex</i> . . . . .	135
6.5	Distribution of the sizes of the labels on <i>EUR-Lex</i> . . . . .	135
6.6	Pseudocode of the incremental training method of the DMLPP algorithm . . . . .	138
6.7	Pseudocode of the prediction phase of the DMLPP algorithm . . . . .	139
7.1	Hierarchy of a HOMER classifier . . . . .	155



7.2	Training time over number of partitions for the six HOMER variants . . . .	162
7.3	Testing time over number of partitions for the six HOMER variants . . . .	162
7.4	Micro recall over number of partitions for the six HOMER variants . . . .	164
7.5	Micro precision over number of partitions for the six HOMER variants . .	164
7.6	Micro F1 over number of partitions for the six HOMER variants . . . . .	165
8.1	Pairwise training on two separate tasks . . . . .	174
8.2	Pairwise training on the global task . . . . .	174
8.3	Calibration in the two separate tasks . . . . .	176
8.4	Calibration in the global task . . . . .	176
8.5	Joined calibration in the global task . . . . .	176
9.1	The LeGo framework . . . . .	184
9.2	Decomposition of multilabel training for BR, MC and LP . . . . .	186
9.3	Example Bayesian networks . . . . .	189
9.4	A MLC problem and its representation in pattern space . . . . .	191
10.1	Transformation of a text sentence into a classification problem . . . . .	203
10.2	Example of a syntax tree . . . . .	204
11.1	Schematic complexity diagram for multilabel classification . . . . .	210

---

## List of Tables

2.1	Classification type matrix . . . . .	15
2.2	Statistics of multilabel datasets . . . . .	44
3.1	Overview of transformation approaches and properties . . . . .	54
3.2	Complexity comparison . . . . .	66
3.3	Extended complexity comparison under idealistic assumptions . . . . .	68
3.4	Example of joint and marginal probabilities . . . . .	75
4.1	Computational complexity of perceptron-based MLPP, BR and MMP . . . . .	94
4.2	Comparison of CMLPP vs. others on the <i>rcv1</i> test set . . . . .	97
4.3	Comparison of the computational costs . . . . .	101
4.4	Comparison of CMLPP vs. others on others test sets . . . . .	106
5.1	Sports competition example . . . . .	112
5.2	Computational complexity comparison of the QVoting approach . . . . .	117
5.3	Statistics of datasets for the QVoting experiments . . . . .	118
5.4	Computational costs comparison to QVoting CLR . . . . .	119
5.5	Bipartitioning performance comparison to QVoting CLR . . . . .	124
5.6	Computational Costs for QCLR with SVMs . . . . .	125
5.7	Bipartitioning performance with SVM as base classifier . . . . .	126
6.1	Statistics of the EUR-Lex datasets . . . . .	132
6.2	Computational complexity of the perceptron based algorithms . . . . .	141
6.3	Average ranking losses on the EUR-Lex datasets . . . . .	144
6.4	Average bipartitioning losses on the EUR-Lex datasets . . . . .	146
6.5	Computational costs on the EUR-Lex datasets . . . . .	147
6.6	Memory requirements for the <i>EUR-Lex</i> datasets . . . . .	149
7.1	Complexity comparison of BR and CLR combined with HOMER . . . . .	158
7.2	Dataset statistics for the HOMER experiments . . . . .	160
7.3	Bipartition measures and computational costs for HOMER . . . . .	166
8.1	Statistics for datasets for parallel tasks experiments: <i>EUR-lex</i> and <i>rcv1</i> . . . . .	177
8.2	Statistics for datasets for parallel tasks experiments: <i>hifind</i> . . . . .	178
8.3	Results of DCMLPP on the <i>EUR-Lex</i> dataset, parallel tasks . . . . .	179
8.4	Results of HOMER on the <i>hifind</i> dataset, parallel tasks . . . . .	180
8.5	Results of MLPP on the <i>rcv1</i> dataset, parallel tasks . . . . .	180



9.1 Datasets used in the subgroup discovery experiments . . . . . 192

9.2 Average ranks of different feature selection methods . . . . . 193

9.3 Comparison of different feature sets . . . . . 195

9.4 Comparison of the different base learners . . . . . 196

9.5 Comparison of the different decomposition approaches . . . . . 197

10.1 Prediction quality comparison for syntactic parsing . . . . . 207



---

# 1 Introduction

It is a human necessity to comprehend and organize the environment. Hence, things in the real world are often analyzed, characterized, described, collected, archived, ordered, grouped and catalogued. One of the most studied objects are writings. Libraries of texts on papyrus in Ancient Egypt are known to have existed more than a thousand years BC. Even in these early antique collections, the writings were ordered or grouped according to certain characteristics. Nowadays, we find books ordered according to authors, titles, publication years, languages etc., and classified into genres, topics, subjects and so on.

These assignments were usually made by humans. The aim of *Machine learning* is to provide tools and algorithms which facilitate an automatic machine-driven assignment. Learning or classification algorithms e.g. *learn* from previously seen assignments of *classes* to objects and aim at using this “experience” for the automatic *classification* of new, unseen objects. Such things or objects in this context are often also referred to as *examples* or *instances*. Objects that are used to learn are commonly called *training* examples, and an automatic classification or *prediction* is done on a *test* example.

A classic exercise in machine learning is *multiclass classification*. The task consists in learning an assignment of objects to one class in a set of alternatives. Returning to our library example, the possible classes could be genre labels such as *fiction*, *literature*, *romance*, *crime* etc. Multiclass classification is very well studied and many solutions have been developed for this problem. In recent years however, a related task has gained increased attention: *Multilabel classification* assumes that classes are not necessarily disjoint and, hence, that a thing can be assigned to multiple classes simultaneously. The given example of the list of genres indeed seems to indicate that this is not an exceptional but rather the usual case.

However, the limited degree of previous research on multiclass problems had very concrete reasons. Firstly, available catalogs often did not allow multiple assignments. This ensured simplicity and hence efficiency (e.g. in locating books) on the one hand and effectivity (no copies needed, see below) on the other. Classification systems with elaborated topic hierarchies were specifically developed in order to prevent multiple assignments. In the exceptional case that a *multilabel* assignment was needed, things were either assigned to the most descriptive, fitting class, or they were just replicated and then put into several classes. If multiple assignments could not be avoided, separate classification systems were defined according to orthogonal facets, with one facet serving as the main one. Secondly, the practical and self-imposed restriction to the multiclass case in the past considerably helped the research community to understand some of the general foundations of learning algorithms. Moreover, the available resources, both in terms of data and computationally, were restricted.

---

With the rise of electronically available information and the “digital revolution of information society”, most of the stated classic limitations have become unnecessary. The vast amount of electronically available data demands for new solutions and approaches since pure human classification has become impractical. A prototypical example for the new challenges and possibilities are the meanwhile ubiquitous participatory networks, the so called *Web 2.0*. They make it easy for a user to *label* “things” with keywords, without restrictions regarding the number of assignments to an item.

Efficient approaches are more than ever necessary in order to tackle this large amount of data. However, previous advances, especially regarding the accuracy of automatic classification, should not be ignored.

One of these notable advances concerns the initial decomposition of the original task. The division into several tasks is a commonly applied technique in order to simplify the original problem so that it becomes amenable to existing learning approaches. The decomposition into one problem for each class is the oldest and simplest approach in machine learning: A *classifier* is learned to distinguish objects of one specific class from objects not belonging to this class, which is why it is usually called *one-against-all*. In *pairwise decomposition* we learn classifiers that are able to distinguish two classes, i.e. whether it belongs to one or to the other.

The approach of *pairwise comparison* was first scientifically investigated in 1927 by [Thurstone](#) in the context of psychology and the measurement of personal feelings and preferences. In machine learning, the first attempts encompass the works of [Knerr et al. \(1990\)](#) and since then the superiority to one-against-all was shown in several studies (cf. Section 3.5.7). Moreover, many machine learning software tools use pairwise decomposition as the default setting, including two popular implementations of the state-of-the-art support vector machines algorithm ([Chang and Lin 2001](#), [Witten and Frank 2005](#)), so that we can expect that its use is wide-spread. The main reason for the advantage over one-against-all is, intuitively, that it is easier to learn to distinguish between two classes than between one and all of the remaining classes.

However, one obvious drawback of pairwise decomposition lies in the fact that a class has to be compared to each other class separately; hence a quadratic number of classifiers, with respect to the number of classes, is necessary. [Fürnkranz \(2002\)](#) showed that, surprisingly, it is more efficient in the multiclass setting to learn this quadratic number of classifiers than the linear number of classifiers needed for one-against-all. Still, applying the pairwise classifiers is more expensive than for the one-against-all approach.

Furthermore, it has not been clear to what extent these statements apply to the more complex multilabel classification setting. In face of the increased demands due to the explosion of available data, it is also not clear how classical techniques, and in particular pairwise learning, will behave. Of course, the hope is to be able to benefit from the important advantages and improvements of the pairwise approach even under these adverse circumstances. The main objective of this work shall thus be to investigate methods and techniques for *efficient pairwise multilabel classification*.

The next section will identify the main challenges in automatic classification and in particular in multilabel classification.

---

## 1.1 Challenges in Pairwise Multilabel Classification

Decomposition, and in particular pairwise decomposition, is a general approach for classification that decomposes a global task into several sub-tasks which have to be solved with conventional algorithms and tools from machine learning. Therefore, the limitations and dependencies of the underlying solvers are commonly shared. In addition, new restrictions arise from the particular method of the pairwise decomposition, namely from the explicit consideration of all possible pairs of classes.

The following enumeration will state the main challenges in pairwise multilabel classification. We will see which problem variables of a multilabel problem may affect scalability, efficiency and predictive quality of the pairwise approach.

Without going into more formal details, we define efficiency as the processing speed in terms of objects per time and scalability as the ability of a learning approach to handle a growing quantity. We will generally consider the predictive quality separately from the aforementioned quantities. Obviously, scalability is bounded by efficiency, since if efficiency decreases with a growing variable, an approach has demonstrated its inability to scale. Thus, in our context of learning and classifying, scalability is mainly concerned with memory issues, since speed is already covered by efficiency.

We will commonly assume, without loss of generality, a task of learning and classification of multilabel texts since text problems frequently cover several of the following issues. However, our considerations are not limited to these fields, and in fact multilabel classification come up in applications as diverse as music classification, image, video and semantic scene classification and protein function classification (cf. Section 2.9.1).

- **Dimensionality of input: the number of features**

In classification, an object is described by a set of features and feature values, also called attributes and attribute values (cf. Section 2.1). A common representation for text e.g. is to indicate for each word (feature) the number of times it appeared in the text (value). Obviously, this could lead to very high number of features for vast document collections. Many learning algorithm are very susceptible to this quantity for different reasons. Decision trees and rule learners e.g. have to explore the space of the features since they rely on patterns of feature values. On the other hand, the classification costs are usually not affected so much, since the number of feature tests is relatively low. For other approach such as linear classifiers (cf. Section 4.1.1) the training costs are less controlled by the number of features, but instead the memory requirements.

Pairwise classification shares these dependencies on the number of features since it decomposes the original problem and has further to apply conventional approaches to the generated subproblems. Fortunately, several methods in machine learning aim at a pre-selection or reduction of features. They provide an effective way of keeping the influence on scalability and efficiency constant, so we will resort to these if necessary.

---

- **Quantity of data: the number of examples**

Particularly in the early days of machine learning, appropriate data for experimentation was costly and hence scarce. Nowadays, we face vast amounts of data, which is mainly reflected in the number of examples a particular database contains. A view on two collections available from the Reuters agency demonstrates this. While the first one, retrieved in 1987, contained a little bit more than 20,000 news articles,<sup>1</sup> the second one already collected more than 800,000 documents in one year (beginning in August 1996, see [Lewis et al. 2004](#)). The Internet has also frequently served as a potentially infinite source for new classification benchmarks since then (cf. Section 2.9.1).

Learning algorithms behave very differently with respect to the number of data provided for training. Some *support vector machines* (cf. Section 4.1.6) implementations e.g. have to compare each training example with all other training examples, which makes them infeasible for the aforementioned cases. *Lazy* techniques also quickly become impractical, since they almost have zero training costs, but they must compare each test example to each training example during prediction. Moreover, both approaches have to maintain at least a part of the training data in memory. Hence, great care has to be taken at the time of selecting the appropriate base classifier and it is absolutely necessary to foresee possible limitations.

The increasing availability of data and hence the need for efficient processing was one of the main starting points of the present work. Since a reduction of the acceleration of data growth cannot be expected, this issue remains a key challenge for the future.

- **Availability of data: real-time processing**

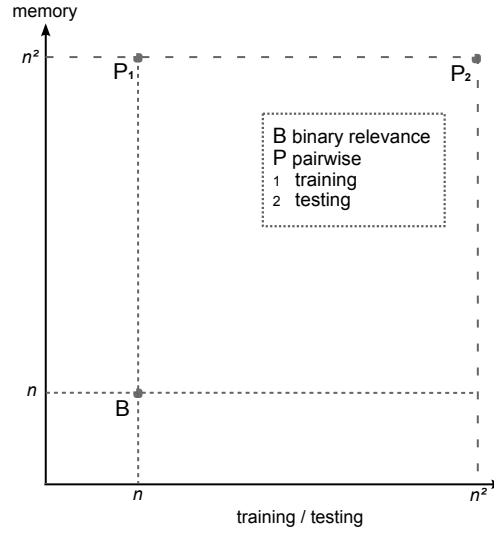
This point may be seen as a direct consequence of the previously discussed explosion of the availability of data. Part of the effort shall hence be dedicated to the investigation of approaches which enable processing of multilabel data with almost no or small delay. This includes the immediate consideration of new training examples for successive predictions as well as the instantaneous classification of a new example in the sense that a prediction is not delayed by the continuous training. In machine learning, these two aspects are typically described as *online* or *incremental* training (cf. e.g. [Sebastiani 2002](#), Sec. 6.6) and *anytime* classification, respectively.

- **Dimensionality of output: the number of classes**

In classification, the *output* usually refers to the prediction of a classifier and the number of alternatives that may be predicted is higher the more classes we have. Therefore, the number of classes is also a variable of great importance to multiclass classification.

---

<sup>1</sup> This is the oldest multilabel dataset used in machine learning research known to the author ([Hayes and Weinstein 1991](#), [Lewis 1992, 2004](#)). In the *20-newsgroups* dataset postings may also belong to several folders, but the dataset was simplified to multiclass by replicating the instances ([Mitchell 1997](#)).



**Figure 1.1:** Schematic complexity diagram comparing pairwise decomposition and one-against-all. Training and testing (x-axis, sub-indices 1 and 2, respectively, are used if not equal), memory (y-axis) are shown with respect to the number of classes  $n$ .

But the output dimensionality plays the key role particularly in pairwise classification, since the number of classifiers which have to be learned quadratically depend on the number of possible classes. A few recent examples emphasize this: a dataset extracted from the social bookmarking page *del.icio.us* had documents assigned to almost 1000 possible tags. And the *EUR-Lex* collection of official documents of the European Union presented in this work is indexed with a taxonomy containing almost 4000 keywords. This would require a number of pairwise classifiers that reaches the magnitude of millions. Without adequate solutions such scenarios would definitively reach the boundaries of scalability and efficiency of the pairwise approach. The **number of classes** is therefore the **main challenge** for pairwise learning.

A schematic comparison of the distinct complexities of one-against-all and pairwise decomposition is shown in Figure 1.1. The diagram shows the dependence of the training and prediction costs (projection on the x-axis) and memory (y-axis) on the number of classes  $n$  for multiclass problems. For multilabel problems, we can expect further movements to the upper-right corner, as we will see in the next paragraph. But the goal shall be to approach the bottom-left corner. A resolution including the solutions in this work will be given at the end of the work in Figure 11.1.

Two additional issues emerges from the fact that the multilabel setting allows assigning an arbitrary number of labels to one instance. Firstly, and this is common to all multilabel solutions, dropping the multiclass restriction suddenly leads to exponential grow of the number of alternatives. Whereas in multiclass classification the output space was the set of classes, it is the power set of the set of classes for

---

multilabel classification. This issue is alleviated by simplifying the task to predicting a ranking over the labels, but this entails further complications, as we will see in this work. Secondly, and this is specific to pairwise learning, the training efficiency is decreased since the costs grow with the possible combinations between true and false labels. In multiclass classification the number of combinations is always constant and approximately the number of classes, but in multilabel data the size of the correct labelset becomes arbitrary and the number of combinations grows exponentially. The present work will also analyze in more detail this efficiency aspect. Note that the number of classifiers, and hence scalability and prediction efficiency, is not touched by this issue.

- **Dependencies between the Labels**

A point which clearly distinguishes multilabel from multiclass problems are the possible dependencies between the classes. Labels can co-occur or correlate with other labels and this might be an indication for certain dependencies. Imagine e.g. a book which was assigned to the keyword *murder*. We could surely suspect that it was also annotated with *crime*. Indeed, this relation could be imperative, but note that the opposite is not as probable.

A frequent concern in multilabel research from the beginning (for an early example, see [McCallum 1999](#)) is hence the modeling and exploitation of this additional data for producing more accurate results. This point will also be covered in this work, but to a smaller degree than our main issue of efficiency and scalability. The concern will rather be to investigate whether and to which degree the pairwise approach can exploit label dependencies under the constraints of high efficiency.

## 1.2 Contributions and Organization of the Work

In light of the stated demands and challenges of current and modern multilabel classification, this work makes the following main contributions:

- A general review of pairwise decomposition in multilabel classification is provided.
- A new multilabel dataset is provided which is currently the most challenging benchmark available in multilabel classification research due to the high dimensionality on the classes.
- A family of pairwise and combining learners is developed that provide the appropriate instruments in order to respond to any combination of the stated challenges and demands on efficiency and scalability.
- The suitability of the frame is substantiated by a detailed formal and empirical investigation of the computational costs of the different decompositive approaches, particularly of the introduced pairwise learners.

---

The following listing shows the organization of the present work. A short summary is given for each chapter presenting further contributions which could not be reflected above in the condensed enumeration of the major contributions. In addition, significant parts of this thesis have previously appeared in publications of the author. The most relevant are accordingly indicated.

- **Chapter 2**

This chapter introduces concepts, notations and the corresponding basic formal definitions required throughout this work. Furthermore, I discuss and review relevant existing and newly introduced evaluation measures, methods of comparison, prior learning approaches and the multilabel data available for research.

- **Chapter 3**

The basic decompositive approaches to multilabel classification are presented in Chapter 3. The formal analysis of the pairwise decomposition approach provides the basis for later analyses in this work. Critical aspects and limitations as well as advantages and disadvantages of the pairwise approach are discussed and summarized.

- **Chapter 4**

This chapter connects to the prior work of the author (Loza Mencía 2006, diploma thesis), which presented the efficient **MLPP** ranking algorithm. MLPP is extended so as to incorporate **calibration** (Brinker et al. 2006), a new technique in the framework of pairwise preference learning which naturally enables to predict a set of labels. An extensive empirical study, the first of its sort, confirms the superiority of the pairwise approach, the suitability of calibration and the ability for real-time processing of a high quantity of data of **CMLPP**.

- Loza Mencía and Fürnkranz 2008c
- Fürnkranz, Hüllermeier, Loza Mencía and Brinker 2008

- **Chapter 5**

CMLPP is naturally combined with the approach of **QVoting** (Park and Fürnkranz 2007) to **QCMLPP** (Loza Mencía et al. 2009), which is able to break the main limitation of the pairwise approach with respect to efficiency, namely the quadratic number of evaluations during prediction. It is demonstrated in a comprehensive empirical evaluation that it is possible to considerably improve this to a log-linear dependency on the number of classes. Moreover, the results show again the superiority over the competing decompositive approaches, regardless of the employed base learner (**QCLR**).

- Loza Mencía, Park and Fürnkranz 2009
- Loza Mencía, Park and Fürnkranz 2010

- **Chapter 6**

A vast collection of legal document from the European Union with a set of roughly



---

4000 classes is introduced in this chapter. Whereas this high number impeded employing pairwise decomposition in a first study (Loza Mencía and Fürnkranz 2007a), MLPP was subsequently reformulated to the mathematically identical **Dual MLPP** which enables scaling pairwise decomposition to the resulting 8 million classifiers. In contrast to MLPP, the pairwise base linear classifiers in DMLPP are represented in the *dual* as linear combination of the training examples instead of explicitly as a decision vector. The training set and the necessary coefficients easily fit in memory in contrast to conventional MLPP. The dual variant substantially outperformed the baselines on the **EUR-Lex** dataset in terms of predictive quality.

- Loza Mencía and Fürnkranz 2007a
- Loza Mencía and Fürnkranz 2008a
- Loza Mencía and Fürnkranz 2010

- **Chapter 7**

The combination of QCLR with a hierarchically decomposing approach named HOMER (Tsoumakas et al. 2008) provides a more than suitable basis for even more challenging cases. HOMER decomposes the original problem into a hierarchy of considerably simpler multilabel problems, where each subproblem corresponds to a meta-label in the parent problem. The subproblems are then solved with QCLR. As it turns out, both approaches harmonize perfectly and the one-against-all baseline is again outperformed in terms of predictive quality, but also in training and testing efficiency, whereas the memory requirements only differ by a constant factor. In contrast to DMLPP, **H-QCLR**'s formulation is not equivalent anymore, but the simplification preserves the positive effects of pairwise classification.

- Tsoumakas, Loza Mencía, Katakis, Park and Fürnkranz 2009b

- **Chapter 8**

This chapter describes a first attempt at exploiting label correlations with pairwise classifiers. It is shown that a globally trained classifier is superior to a locally trained one in a multilabel multi-task setting, which demonstrates the ability to exploit inter-label connections. The advances in the previous chapters prepared the ground for these experiments, since the multi-task setting multiplies the number of labels in the global problem.

- Loza Mencía 2010b

- **Chapter 9**

The issue of label dependencies is also the focus of this chapter, which describes a first investigation in order to connect conventional multilabel classification to local pattern discovery. Locally exceptional patterns in the labels are extracted and an empirical study shows the benefit of this approach, not only for pairwise decomposition.

- Duivesteijn, Loza Mencía, Fürnkranz and Knobbe 2012



---

- **Chapter 10**

The work described in this chapter is related to Chapter 8 but has information extraction and the common approach to learn each type of information separately as major concern.

- [Loza Mencía 2010a](#)

- **Chapter 11**

The last chapter summarizes the findings and results of the previous chapters. The big picture is drawn and the perspectives for possible future works are shown.



---

## 2 Fundamentals of Multilabel Classification

Before we focus on the problem of multilabel classification, we have to state the general setting of a classification task in machine learning (Sections 2.1 to 2.4). The multilabel scenario is then demarcated from related types of problems in Section 2.5. Section 2.7.3 and 2.10 are dedicated to the evaluation and comparison of the prediction quality of multilabel classifiers, which we can train on the datasets presented in Section 2.9.1.

### 2.1 Input Object Space

In the field of machine learning, *classification* denotes the task of learning an association of *objects* to *classes* in a supervised manner. An object may be anything, a document, an image, person, etc. In order to being able to process the objects, we require that the object can be represented by a list of characteristics or attributes.<sup>2</sup> In particular, we will represent an instance or object as a vector  $\mathbf{x}$  of real-valued attribute or feature values in a feature space  $\mathcal{X}$

$$\mathbf{x} = (x_1, \dots, x_a) \quad \mathbf{x} \in \mathcal{X} \quad \mathcal{X} \subseteq \mathbb{R}^a \quad (2.1)$$

with  $a$  as number of features. From the popular perspective of probability theory,  $\mathbf{x}$  is often considered an independent and identically distributed random variable drawn from a fixed but unknown distribution over  $\mathcal{X}$ .

We will commonly not distinguish between input and feature space although they might not always be identical.<sup>3</sup>

### 2.2 Classification Mapping

Each instance  $\mathbf{x}$  is related to one or several classes. A class may represent any arbitrary characteristic of an object. However, often it denotes a particular category an object may

---

<sup>2</sup> This is in fact specifically a requirement for classification. In object ranking e.g., roughly speaking the task of mapping from a user to a ranking of documents in dependency of a query, it may not be necessary to provide a description of the user when representations of the documents are available.

<sup>3</sup> An instance may contain features that are not directly representable in  $\mathbb{R}$  without prior transformation, e.g. nominal attributes. We may say that the original instance is given in the input space before being transformed into a feature space vector. Similarly if kernels are used the input space is different from the feature space, since the instance vectors are implicitly transformed into the higher dimensional feature space (cf. Section 4.1.6).

belong to, which is why classification is also often called *categorization*, especially when the objects are text documents. Another often used term, especially in the context of multilabel classification, is *labels* instead of *classes* (cf. Section 2.5.3). Formally we denote a class association as a relation  $f$  between the input space  $\mathcal{X}$  and the output space  $\mathcal{Y}$

$$f : \mathcal{X} \times \mathcal{Y} \rightarrow \{\text{true}, \text{false}\} \quad (2.2)$$

Under the assumption that an object  $\mathbf{x}$  and its representation always has the same class or classes associated and also that this mapping does not change over time,<sup>4</sup> we can define the relation as a function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  of  $\mathbf{x}$ , which is commonly neither injective nor necessarily surjective

$$\mathbf{y} := f(\mathbf{x}), \mathbf{y} \in \mathcal{Y} \quad \Leftrightarrow \quad \text{object } \mathbf{x} \text{ is mapped to } \mathbf{y} \quad (2.3)$$

## 2.3 Learning a Model and Predicting

A learning algorithm for classification is an algorithm that tries to learn the mapping between objects and classes from a set of given exemplary mappings. More specifically, it learns a *hypothesis*, *model*, *predictor* or *classifier function*

$$h : \mathcal{X} \rightarrow \mathcal{Y}$$

which aims at behaving like the original mapping  $f$ . The function  $h$  is *learned from*, *induced from* or *trained on* a sequence of given mappings, which we formulate as

$$\mathcal{T}_{rain} := \langle (\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_{|\mathcal{T}_{rain}|}, \mathbf{y}_{|\mathcal{T}_{rain}|}) \rangle \quad (2.4)$$

and we may write  $h_{\mathcal{T}_{rain}}$  to specify on which data a model was learned. Especially in *batch learning* this sequence is often defined as a set and hence called *training set*, although real training data may contain duplicates and the ordering of the elements may be relevant for the learning algorithms. We shall only distinguish between both definitions if it is relevant for the current setting.

The outcome

$$\hat{\mathbf{y}} := h(\mathbf{x}) \quad \hat{\mathbf{y}} \in \mathcal{Y} \quad (2.5)$$

is referred to as the *prediction* of the classifier and  $\mathbf{x}$  is called a *test example* in this context. A classifier is hence usually evaluated on a *test set* measuring the discrepancy between

<sup>4</sup> Note that this cannot be guaranteed in practice: noisy data or insufficient characterization due to e.g. too strong feature subset selection can lead to several non identical objects with equal feature vectors. Several approaches exist for dealing with this case. In this work we will usually just accept that inconsistent data may exist and assume a robust processing of the underlying algorithm.

the predictions and the true mappings. Without loss of generality, we define the test set as a sequence of test examples following the sequence of training examples:

$$\mathcal{T}_{est} := \langle (\mathbf{x}_{|\mathcal{T}_{rain}|+1}, \mathbf{y}_{|\mathcal{T}_{rain}|+1}), \dots, (\mathbf{x}_{|\mathcal{T}_{rain}|+|\mathcal{T}_{est}|}, \mathbf{y}_{|\mathcal{T}_{rain}|+|\mathcal{T}_{est}|}) \rangle \quad (2.6)$$

It is important to note that although we have given the true mappings  $\mathbf{y}_i$  in Eq. 2.6, they are *unknown* and not available to the learning algorithm. Hence, only  $\langle \mathbf{x}_{|\mathcal{T}_{rain}|+1}, \dots, \mathbf{x}_{|\mathcal{T}_{rain}|+|\mathcal{T}_{est}|} \rangle$  is presented to the algorithm.

Usually we are interested in evaluating the induction ability of a learning algorithm, i.e. the ability of inducing from the training set  $\mathcal{T}_{rain}$  the correct, true classification  $\mathbf{y}$  for a (test) example  $\mathbf{x}$ , which was previously not seen or known and for which the only available information is that it was presumably i.i.d. drawn from the same distribution as the objects in  $\mathcal{T}_{rain}$ . Therefore, it is very important to ensure that the intersection between both sets is empty,

$$\mathcal{T}_{rain} \cap \mathcal{T}_{est} = \emptyset \quad (2.7)$$

i.e. that no training examples are used to evaluate a classifier.<sup>5</sup>

The difference between the true assignment  $\mathbf{y}$  and the predicted output  $\hat{\mathbf{y}}$  is measured by an error function which receives the true and a predicted output<sup>6</sup>

$$\delta : \mathcal{Y} \times \mathcal{Y} \rightarrow [0; \infty) \subset \mathbb{R} \quad \forall \mathbf{y} \in \mathcal{Y}. \quad \delta(\mathbf{y}, \mathbf{y}) := 0 \quad (2.8)$$

and which is usually selected according to the particular problem setting and user demands, see also Section 2.7 for the options.<sup>7</sup> A popular approach is to define the process of learning the mapping  $f$  as finding a *risk-minimizing* classifier  $h^*$  (e.g. Dembczyński et al. 2010a,c). Such model minimizes the expected loss of a classifier  $h$  over the joint distribution  $P$  of the input and output spaces  $\mathcal{X}$  and  $\mathcal{Y}$ , i.e. it is given by

$$h^* := \arg \min_h \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim P} \delta(\mathbf{y}, h(\mathbf{x})) \quad (2.9)$$

In other words, if we are able to choose our classifier in this way, it should minimize the error on our test set, since the test examples are drawn from  $P$ . A discussion on implicitly and explicitly minimizing multilabel measures and generally good working classifiers can be found in Section 2.7.5.

<sup>5</sup> On the other hand, evaluating the classifier on examples from the training set corresponds to evaluating the consistency degree of a learning algorithm. An algorithm is declared *consistent* on a (training) set  $\mathcal{T}$  if  $\forall \mathbf{x} \in \mathcal{T}. \quad h_{\mathcal{T}}(\mathbf{x}) = f(\mathbf{x})$  holds. Another purpose of evaluating on the training data is to see if the data is *separable* with respect to a specific (class of) classifier(s), e.g. linear classifiers (cf. Section 4.1.1).

<sup>6</sup> We restrict at this point our focus, with some loss of generality, on error functions that evaluate *per example* in contrast to possible errors that may be only computed on example sets:  $\delta : \mathcal{Y}^{|\mathcal{T}|} \times \mathcal{Y}^{|\mathcal{T}|} \rightarrow [0; \infty)$ . See also Section 2.7 and Section 2.7.1.

<sup>7</sup> Many existing measures do not adhere to Eq. 2.8 in their original definition since they are e.g. formulated as quality function where greater values are better. However, every measure can be reformulated as error function conforming to Eq. 2.8

## 2.4 Output Label Space

Before we further define the output  $\mathbf{y}$  and output space  $\mathcal{Y}$ , we will analyze class associations from a more intuitive set-theoretic point of view. Note that the following formulations give a generic definition of classification which will be further restricted appropriately depending on the considered subproblem.

The finite set of  $n$  classes an object might be associated to is defined as follows:

$$\mathcal{L} := \{\lambda_1, \dots, \lambda_n\} \quad n = |\mathcal{L}| \quad (2.10)$$

The set of classes  $P$  a particular object is actually associated to has the form

$$P \subseteq \mathcal{L} \quad N := \mathcal{L} \setminus P \quad P, N \in 2^{\mathcal{L}} \quad (2.11)$$

so that  $P \cup N = \mathcal{L}$  and  $P \cap N = \emptyset$ .  $2^{\mathcal{A}}$  shall symbolize the powerset of a set  $\mathcal{A}$ . We denote the classes in  $P$  as *positive* or *relevant*, and the classes in  $N$  as *negative* or *irrelevant*, respectively. Especially in the context of multilabel classification, we denote  $P$  as the positive or relevant *labelset* associated to an example. We shall differentiate it from the set of possible classes or labels  $\mathcal{L}$ . If it is clear to which arbitrary or particular object  $\mathbf{x}$  we refer, we omit the indices and write e.g.  $P$  instead of  $P_{\mathbf{x}}$  such as in Eq. 2.11.

Given the definitions of  $\mathcal{L}$  and  $P$ , we define the  $n$ -dimensional vector output space  $\mathcal{Y}$  of a  $n$ -classes problem as

$$\mathcal{Y}^n := \{0, 1\}^n \quad (2.12)$$

The values 1 and 0 were chosen arbitrarily for the sake of formal simplicity in following definitions, but any other two-elements set  $\mathcal{A}$  with  $|\mathcal{A}| = 2$  and an (total) order on both elements is valid. A different very common representation is e.g.  $\{-1, 1\}$ . In both cases 1 denotes the presence of an association,  $\{-, +\}$  would be hence a very appropriate symbolic representation. If  $n$  is fixed and clear, we shall omit the index and write only  $\mathcal{Y}$ .

We further define the allocation of an output vector  $\mathbf{y}$  as

$$\mathbf{y} = (y_1, \dots, y_n) \in \mathcal{Y} \quad y_i := \begin{cases} 1 & \text{if } \lambda_i \in P \text{ is relevant} \\ 0 & \text{otherwise (if } \lambda_i \in N) \end{cases} \quad (2.13)$$

as an additional specification to Eq. 2.3.

The definition of the mapping from objects to classes from a set-theoretic and from a vector representational perspective allows simpler, more intuitive and more adjusted formulations depending on the actual situation. We will often include both representations throughout this work.

**Table 2.1:** Classification type matrix depending on cardinality and dimension of the problem setting. The columns denote the dimensions, the rows the cardinality.  $|\mathcal{L}| = |P| = 1$  is not possible, cf. p. 16 in Section 2.5.1.

cardinality \ dimension	$ \mathcal{L}  = 1$	$ \mathcal{L}  > 1$
$ P  = 1$	–	multi-class classification
$0 \leq  P  \leq  \mathcal{L} $	binary classification	multilabel classification

## 2.5 Types of Classification Problems

The type of classification problem is determined by the characteristics of the output space  $\mathcal{Y}$ . We distinguish mainly between two properties, the *dimension* of the output space and the *cardinality* of the mappings, and particularly whether they are different from one. The number of dimensions, i.e. the number of classes, determines whether we are addressing a *binary* or a *multiclass* problem, *single-label* or *multilabel* classification depends on the cardinality.<sup>8</sup> If additionally there exists a partially ordering relation on the classes resulting in a rooted tree over the classes, then we call the problem *hierarchical*.

The *cardinality* of a labelset  $P$  or an output vector  $\mathbf{y}$  of a document  $\mathbf{x}$  is defined as

$$|\mathbf{y}| := |P| = \sum_{1 \leq i \leq n} y_i \quad (2.14)$$

with the operator  $|\cdot|$  applied on a set counting the number of elements contained. If we only allow a cardinality of one for every possible instance, then the problem is called *single-label*, otherwise it is called a *multilabel* problem. A *binary* problem is given when the set of classes only contains one element, otherwise it is called *multiclass*.

An overview of the different combinations of cardinality and dimensions is given in Figure 2.1. The different resulting types of classifications are worked out and formalized in more detail in the following.

### 2.5.1 Binary Classification

In binary classification, an instance is associated with one of two possible distinct outputs. An instance is hence associated to a *binary variable* or *binary class*. The binary class may be e.g. a certain property of the object that shall be either present or absent.

This example or setting is commonly also referred to as *concept learning*, which is dedicated to infer a model or description of a target concept from specific examples of it (see e.g. Domingos 1997, Sec. 2.2, Mitchell 1997, Ch. 2). Based on this perspective of the problem, an instance, for which the characteristic is present, is hence called a *positive*

<sup>8</sup> We emphasize the high relevance in machine learning of "multilabel" (especially in this work) and "multiclass" classification by employing the closed compound form of the terms, which is often adopted when a concept has been established. Compare the accepted "online" to "on-line" or the original "on line".

example. A *negative example* is consequently an instance for which the characteristic is not true. It is therefore common to denominate this task a two-class classification task,<sup>9</sup> since we map objects to the *positive class* or to the *negative class*.

Concept learning focuses on one side, the positive class, for which it assumes a clear semantic. This is not done for the opposite *non-concept* side, hence learning algorithms used traditionally in concept learning are mainly interested in finding convenient representations of *the concept*. In contrast, binary classification as the general setting does not assume asymmetry, therefore it does also not assume any particular meaning of the possible outputs. However, it is common for practical purposes to differentiate between positive and negative though the determination of one of the possible outputs as positive is formally aleatory.

In fact, the paradigm of learning by pairwise comparison (cf. Section 3.4), on which this thesis mainly builds on, clearly deviates from the semantic of positive and negative used in concept learning. As an example we may see the task of identifying the color of objects. Whereas in traditional concept learning we would learn to detect whether an object is *red* or *not red*, in pairwise learning the target would be to differentiate between *red* and, for instance, *blue* objects. It is important to note this also in context of the following definition. More comments on concept learning with respect to multilabel learning can be found in Section 3.1.

Formally we define binary classification as fulfilling the following property:

$$|\mathcal{L}| = 1 \qquad \mathcal{Y}_{bin} := \mathcal{Y}^1 \qquad (\text{binary class}) \qquad (2.15)$$

That means that we actually formally define binary classification as only being concerned with exactly one class, the positive class  $\lambda_1$ . However, informally we may say that an object belongs to the negative class if it is associated with the empty labelset  $P = \emptyset$ . Put differently, the term two-class problem refers to the two possible mappings  $P = \{\lambda_1\}$  and  $P = \emptyset$ .

The differentiation of cardinality does not consistently apply to binary classification since it is neither single-label nor actually multilabel: an example must not belong to the positive and negative class simultaneously (multilabel), on the other hand the cardinality of an output vector  $\mathbf{y}$  is not always one (single-label). Note again that it would be possible to formalize binary classification as a multiclass problem with two classes, but we prefer the given definition since it is more coherent with the set-theoretic definition and simplifies the transfer from multiclass to binary.

The two-class perspective would naturally allow the very rare *multilabel binary setting*, i.e. where it is desired or required that an example is allowed additionally to belong to both classes simultaneously or to none (e.g. Angulo et al. 2006). We would address this setting as a multilabel problem with two classes, which has indeed formally the same form.

Binary classification is one of the classical problems in machine learning. Separate and conquer rule induction learners (Fürnkranz et al. 2012) are representative algorithms

<sup>9</sup> Another used term is *dichotomous classification*.



particularly used for concept learning tasks, whereas linear classifier algorithms (cf. Section 4.1.1) represent a category of more general approaches. Since binary classification tasks are comparably simple and straightforward, non-binary problems are very often explicitly or implicitly transformed into binary problems, as we will see in Chapter 3.

### 2.5.2 Multiclass Classification

Another classical setting in machine learning is multiclass classification. Most of the datasets in the widely used UCI Machine Learning repository (Frank and Asuncion 2010) are of this type. Many classical algorithms like Naive Bayes and decision tree learner (see Mitchell 1997, Sec. 6.9 & 3) solve specifically multiclass problems.

Indeed, usually *multiclass classification* refers to *single-label multiclass* classification, i.e. we define it for a fixed  $n$  as

$$|\mathcal{L}| = n > 1 \quad \mathcal{Y}_{mc}^n \subseteq \mathcal{Y}^n \quad (\text{multiclass}) \quad (2.16)$$

$$\forall \mathbf{x}. |P_{\mathbf{x}}| = 1 \quad \mathcal{Y}_{mc}^n = \{\mathbf{y} \in \mathcal{Y}^n \mid |\mathbf{y}| = 1\} \quad (\text{single-label}) \quad (2.17)$$

This means that the factual output space is restricted to the subspace of  $\mathcal{Y}^n$  for which exactly one arbitrary dimension  $y_i$  is unequal 0. Combinatorially, this restricts the number of possible labelsets  $\{P_{\mathbf{x}}\} \in 2^{\mathcal{L}}$  to the number of available classes  $n$ .

### 2.5.3 Multilabel Classification

Multilabel classification (MLC) problems have gained increasing attention in recent times (cf. e.g. the workshops organized recently by Tsoumakas et al. 2009c, Zhang et al. 2010a). It denotes the setting in which it is possible to assign several classes to an object. Since the classes can be attached arbitrarily to an object, they are preferably called labels in this context. Other names for the problem scenario include *multi-topic categorization*.

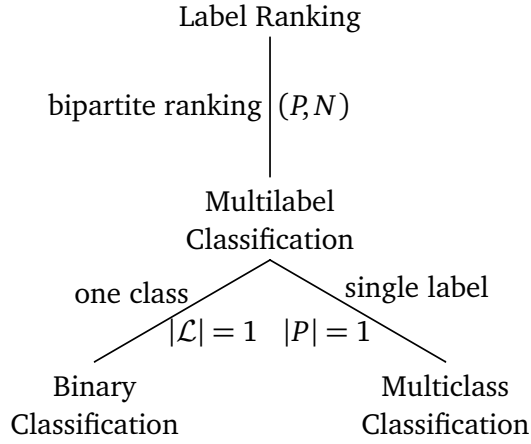
According to our definitions in Section 2.5, the longer correct denomination of the problem setting is *multilabel multiclass classification* and we define it for a fixed  $n$  as

$$|\mathcal{L}| = n > 1 \quad \mathcal{Y}_{ml}^n \subseteq \mathcal{Y}^n \quad (\text{multiclass}) \quad (2.18)$$

$$\forall \mathbf{x}. 0 \leq |P_{\mathbf{x}}| \leq n \quad \mathcal{Y}_{ml}^n := \mathcal{Y}^n \quad (\text{multilabel}) \quad (2.19)$$

Potentially, there are  $2^n$  different allowed allocations of  $\mathbf{y}$  or  $P$ , which is a dramatic growth compared to the  $n$  possible states in the multiclass setting. This, and especially the resulting correlations and dependencies between the labels in  $\mathcal{L}$ , make the multilabel setting particularly challenging and interesting compared to the classical field of binary and multiclass classification.

The multilabel setting can be seen as a general framework for any type of ordinary classification problems. In fact, binary classification can be considered a special case of



**Figure 2.1:** Taxonomy of problem settings from general to special. The descriptions on the edges present informally and formally the restriction on the parent setting which characterizes and specifies the child problem.

multilabel classification for which the number of classes is exactly one, and multiclass classification is a specialization for which the size of the labelsets is fixed to one. In other words, since binary and multiclass classification problems are *contained* in the space of multilabel problems, they particularly can be directly represented as multilabel problems and hence be solved with multilabel algorithms. However, the mapping of multilabel predictions back to multiclass may not be trivial if the predicted labelset does erroneously not contain exactly one label. In this case the prediction can simply be considered as wrong,<sup>10</sup> though in practice an order on the label can often be additionally induced (see next sections). Approaches for solving it in the opposite direction, i.e. transforming multilabel to binary or multiclass problems, are possible and are described in full detail in Chapter 3. Other authors take this possibility as an argument to define binary classification as a generalization of multilabel classification (cf. [Sebastiani 2002](#), Sec. 2.2). However, from the point of view of abstraction, a *solvable-by* relation is not enough to assume subsumption, since this does not imply that binary classification *contains* the multilabel classification problem setting. Moreover, there is no single, direct, imperative way of transforming multilabel problems to binary problems. In fact, there are several different possibilities, which additionally weakens this opposite view.

As can be seen from the visualization of the relationships in Figure 2.1, there is additionally the concept of label ranking that *subsumes* all previously described settings and which is described in the following Section 2.5.4.

<sup>10</sup> However, in the multiclass settings there exist sometimes the option to abstain from predicting ("reject" or "none of the above" option) or to provide a set of candidates (non-deterministic classification, cf. [del Coz et al. 2009](#)). In such settings the formally "wrong" output would be in line again and an evaluation would be possible.

### 2.5.4 Label Ranking

The task of a classifier is to select between several possible predictions. In multiclass classification for example the alternatives are constituted by the set of classes. To this end, most of them compute internally an ordered list over the options, a *ranking*. Especially the output of probabilistic classifiers, that aim at producing valid probabilistic estimations, is clearly a ranking. Even discrete predictions of so called soft classifiers are often accompanied by (continuous) scores, which can be interpreted as confidences.

A ranking over the classes is interpretable in a clear way, and as such it is straightforward in binary and multiclass classification to reconstruct an output in the original discrete output space  $\mathcal{Y}$ , namely by predicting the top ranked class. Moreover, a ranking often allows a better evaluation of the performance of a classifier since it is not only possible to determine whether the correct class was missed but also to which degree. The position error used in multiclass classification e.g., which is closely related to the margin loss in MLC (cf. Section 2.7.4), computes the deviation of the correct class from the top position.

Hence, label ranking in the context of multiclass classification is often defined as the task of predicting a ranking on the set of classes but given a mapping from objects to single classes as training input. We will resolve this asymmetric problem statement further below.

But first, let us define a *total ranking*  $\mathbf{r}$  as a permutation over the class indices  $\mathbb{N}_{\leq n}$ :

$$\begin{aligned}\mathbf{r} &:= \langle \pi_1, \dots, \pi_n \rangle \in \Pi^n \subset (\mathbb{N}_{\leq n})^n \\ \mathbb{N}_{\leq n} &:= \{i \in \mathbb{N} \mid 1 \leq i \leq n\} \\ \Pi^n &:= \{\mathbf{r} \mid \{\pi_1, \dots, \pi_n\} = \mathbb{N}_{\leq n}\}\end{aligned}\tag{2.20}$$

For the sake of simplicity, we allow the following overload for the function  $r$  defined on the ranking  $\mathbf{r}$  which returns the position of a certain label, or the inverse respectively:

$$\begin{aligned}r^{-1}(p) &:= \lambda_{\pi_p} \\ r(\lambda_i) = r(i) = r_i &:= p \iff \lambda_i = r^{-1}(p)\end{aligned}\tag{2.21}$$

Given the ranking  $\mathbf{r}$ , we can formulate a binary order relation  $\succ_{\mathbf{r}}$  on the labels in  $\mathcal{L}$  as

$$\forall 1 \leq i \neq j \leq n. \quad \lambda_i \succ_{\mathbf{r}} \lambda_j \iff r(\lambda_i) > r(\lambda_j)\tag{2.22}$$

which is strict and total, i.e. irreflexive, transitive and all pairs of classes are comparable. In other words, it defines a complete ranking with no ties  $\lambda_{\pi_1} \succ_{\mathbf{r}} \dots \succ_{\mathbf{r}} \lambda_{\pi_n}$  on  $\mathcal{L}$ . This order can be interpreted as a preference relation, i.e. a class is preferred over another class if it is ranked above. Label ranking is hence strongly related to *preference learning* (see Fürnkranz and Hüllermeier 2010, Hüllermeier et al. 2008) and we will often see the connection throughout this work (cf. Section 3.4.2).

In order to model multiclass classification in the framework of label ranking, we need the possibility to express that one class is ranked (i.e. preferred) over the group of all the remaining classes, which in contrast do not have to be ordered. We achieve this by extending our framework in order to allow multi-partite rankings, i.e. we rank (non-overlapping) sets of labels rather than the individual labels. We refer to this setting as *multi-partite label ranking* and consequently slightly reformulate Eq. 2.20 as

$$\begin{aligned} \mathbf{r} &:= \langle \pi_1, \dots, \pi_{|\mathbf{r}|} \rangle \in \Pi_{1,n}^n \\ \pi &\subset \mathbb{N}_{\leq n} \\ \Pi_{u,v}^n &:= \left\{ \mathbf{r} \mid u \leq |\mathbf{r}| \leq v, \bigcap_{1 \leq i \leq |\mathbf{r}|} \pi_i = \mathbb{N}_{\leq n}, \bigcup_{1 \leq i < j \leq |\mathbf{r}|} (\pi_i \cap \pi_j) = \emptyset \right\} \end{aligned} \quad (2.23)$$

with  $\Pi_{u,v}^n$  as the ordered partition of the indices set  $\mathbb{N}_{\leq n}$  with minimum  $u$  and maximum number  $v$  of parts, and the ranking function as

$$\begin{aligned} r^{-1}(p) &:= \{ \lambda_i \mid i \in \pi_p \} \\ r(\lambda_i) = r(i) = r_i &:= p \iff \lambda_i \in r^{-1}(p) \end{aligned} \quad (2.24)$$

though we may use the previous notation if applied to a total ranking for which  $|\pi| = 1$  holds. With respect to the preference relation in Eq. 2.22 we do not have to change the formulation, however we lose the totality property, i.e. there may be pairs of classes which are not comparable.

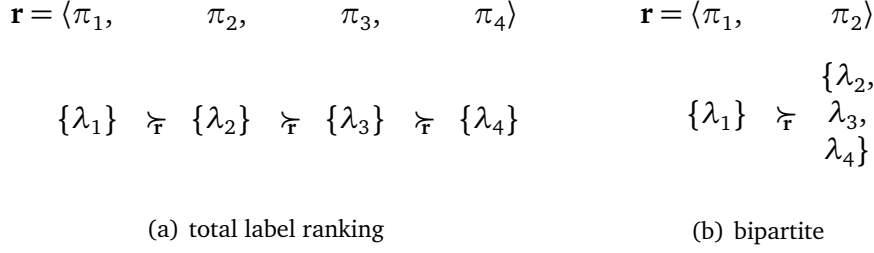
Such a relation defines a strict weak ordering, which corresponds to a ranking with ties if we interpret incomparability as tie, and results in the desired higher-level strict total ordering  $\{ \lambda_i \mid i \in \pi_1 \} \succsim \dots \succsim \{ \lambda_i \mid i \in \pi_{|\mathbf{r}|} \}$  over a partition  $\mathbf{r}$  of  $\mathcal{L}$ .<sup>11</sup>

Multiclass classification can now be instantiated as a label ranking problem with  $|\mathbf{r}| = 2$  and  $|\pi_1| = 1$ . Analogously, we specify multilabel classification with the restriction  $|\mathbf{r}| = 2$ , i.e. *bipartite label ranking*, and set for both  $\pi_1 = P$  and  $\pi_2 = N$ . The case of *total label ranking*, which is commonly simply denominated label ranking, is present when  $|\mathbf{r}| = n$ , i.e. we have  $|\pi_i| = 1$  for all  $1 \leq i \leq n$ . The illustration in Figure 2.2 demonstrates these specializations of multi-partite label ranking.

Hence, based on Eq. 2.3, we can define a multiclass or multilabel classification problem as a mapping function

$$f : \Pi_{2,2}^n \rightarrow \Pi_{2,2}^n \quad (2.25)$$

<sup>11</sup> We could obtain the same meta ordering with a presumably more direct interpretation of ties by giving up the strictness of  $\succ$  and defining a total preorder (non-strict weak order)  $\succeq$  on  $\mathcal{L}$ . However, this would lead to the interpretation that two classes  $\lambda_i, \lambda_j$  contained in a part  $\pi_p$  are equally ranked or preferred, i.e.  $\lambda_i \succeq_r \lambda_j \wedge \lambda_j \succeq_r \lambda_i$ , which is even less accurate, since we usually have the case that we do not know anything about preferring  $\lambda_i$  or  $\lambda_j$  (in multiclass classification e.g. we only know that one class is preferred over all the remaining classes, but not whether or how the remaining classes are ordered), and this is exactly covered by incomparability. See also in this context the discussions and clarifications on the differences between indifference/conflict and incomparability/ignorance in the field of fuzzy preference learning (cf. e.g. Hühn and Hüllermeier 2009, Hüllermeier and Brinker 2008).



**Figure 2.2:** Examples of different instantiations of multipartite label ranking. In the left figure (a), an exemplary total ranking over all classes  $\mathcal{L} = \{\lambda_1, \dots, \lambda_4\}$  is given,  $|\mathbf{r}| = n$ ,  $\pi_i = i$ . In the right figure (b), we can observe a label ranking example with two parts  $|\mathbf{r}| = 2$ , such as in multiclass or multilabel classification. The  $\pi_1$  indices correspond to  $P$  and  $\pi_2$  to  $N$ . The relation  $\succ_r$  symbolizes that all classes somewhere on the left are pairwise preferred over all classes somewhere on the right, hence we obtain 10 preferences on the left and 3 on the right figure.

with the additional constraint for multiclass associations that  $|\pi_1| = 1$ .

However, as already mentioned in the beginning of this section, many learning algorithms learn a function  $h : \Pi_{2,2}^n \rightarrow \Pi_{n,n}^n$  from a bipartition to a total ranking. A function or technique that transforms a full ranking into a bipartite ranking is called *bipartitioning*, or depending on the approach used, *thresholding* or *calibration*, and is given by

$$b : \Pi_{n,n}^n \rightarrow \Pi_{2,2}^n \quad (2.26)$$

For multiclass classification, defining such a function is trivial,  $b(\mathbf{r}) = \langle \pi_1, \bigcup_2^n \pi_i \rangle$ ,<sup>12</sup> leading to the concatenated classifier  $b \circ h$ . However, for MLC it is a much more complex problem that will be addressed in more detail in Section 2.5.5 discussing also the advantages and disadvantages of such a, at first sight, detour.

### 2.5.5 Multilabel Label Ranking

For MLC the advantages of producing a ranking rather than predicting a labelset are not directly obvious. The size of the output space is increased from  $2^n$  to  $n!$  and reconstructing a labelset from a ranking is not as straightforward as for multiclass classification. Regarding the first point, just as the complexity of labelset prediction can be reduced in practice by omitting very infrequent cases, many rankings are indeed equivalent, especially considering the evaluation measure used. For a user<sup>13</sup> e.g. it is usually irrelevant which of two correct labels is ranked above or below as long as they are on top of the negative labels. In fact all of the ranking evaluation measures introduced in Section 2.7.4

<sup>12</sup> Note that Eq. 2.23 does explicitly not exclude the empty set for  $\pi_1$  or  $\pi_2$  in  $\Pi_{2,2}^n$ , i.e. empty labelsets  $|P| = \emptyset$  in MLC are covered by this representation.

<sup>13</sup> We define a user informally as something or somebody using a learning algorithm, i.e. training classifiers and consuming predictions. Usually we can assume a human as the user without loss of generality.

at most consider the order of the positive and negative labels without regarding their particular indices. In other words, the observed rankings are simply sequences of  $+$  and  $-$ , see also Figure 2.3. We can clearly see that this transforms the problem to ranking relevant over irrelevant labels, and that there are  $\binom{n}{|P|}$  different possible sequences. So, if we grossly simplify and suppose that the size of the output space is determined by the ratio of correct and incorrect to correct predictions, we obtain a size of  $\leq n^2/4$ .

Label ranking is therefore often reduced to estimating the relevance of the individual labels in a way that relevant labels ideally obtain higher relevance scores than irrelevant ones. This is a well studied and often used approach in multiclass classification and this makes it simple to apply and benefit from such techniques in MLC. In fact, many MLC approaches are prototype based (cf. Section 3.1) or use generative class models (cf. Section 2.8.2). Recent works of Dembczyński et al. (2010a,c) indicate that concentrating on individual class predictions is potentially sufficient in order to minimize certain losses, especially if the labels are independent (cf. Section 3.5.5).

In particular fields of application, especially in those in which a full automation is not expected but a learning system is comprehended as supportive, it is often sufficient to present a ranking of labels from which a user then selects the appropriate ones. Because of this and because of the outlined advantages, MLC is often simplified to the prediction of label rankings, refer to Elisseff and Weston (2001), MMP (Crammer and Singer 2002, 2003) in Section 4.3 and also basic MLPP in Chapter 4 and 6. Notably, a separation of predicted positive and negative labels can easily be produced by splitting the ranking at a certain position: all labels above the split are considered positive and all below negative. This approach is called *thresholding* or more generally *bipartitioning* and is detailed in Section 3.5.4.

### 2.5.6 Hierarchical Classification

We define a hierarchy as a non-strict partial order over a set of classes that results in a rooted directed tree (arborescence graph). The relation may have a subsumptive or compositional semantic, i.e. child nodes are either a specialization of their parents classes (*is-a* relation) or they are part of their parents (inverted *composed-of* relation).

In hierarchical multiclass classification (cf. e.g. Cai and Hofmann 2004) such a hierarchical ordering  $\mathcal{H}$  on  $\mathcal{L}$  is always pre-specified and only implicitly present in the actual data, since only one class (node) can be assigned to each object. No distinction between flat and hierarchical data is possible. In *hierarchical multilabel classification* (HMLC) the particular order is usually reflected in the data, i.e. all labels between the most specific one and the root node (all parents) are set as relevant. Of course, several most specific labels can be assigned in HMLC. Hybrid settings as by Wang et al. (2011) allow only one path to be set.

Specialized learning algorithms take the hierarchical structure into account, by adapting their problem decomposition or learning strategy, by the organization of the hypothesis space, or by specifically minimizing hierarchical losses. Although much of the available data is inherently hierarchical, this additional information is often ignored and the prob-

lems are usually considered to have a *flat* structure in MLC for the sake of fundamental research (cf. Section 2.9.1 and citations therein). However, upcoming social tagging techniques usually assume flat labels (cf. Section 2.9) and hence this strategy has gained more than ever its right to exist. Moreover, firstly, hierarchical relations in the output space are often not in accordance with (spatial closeness in) the input space (Fürnkranz and Sima 2010, Sima 2008, Zimek et al. 2010). Secondly, inconsistencies between the specified output space and the actual output vectors may appear, which is why some recent approaches in HMLC move on to extracting the hierarchy from the data (Brucker et al. 2011b, Zimek et al. 2010). Thirdly, though there is evidence that exploiting the hierarchical structure has advantages over the flat approach (e.g. Cesa-Bianchi et al. 2006, Vens et al. 2008), Zimek et al. (2010) claim that if a strong flat classification algorithm is used the lead vanishes. This could be e.g. a learner that does implicitly take the order on  $\mathcal{L}$  into account by directly exploiting any type of label dependencies (cf. Section 2.6). Lastly, from a practical point of view, the comparison of flat predictions are much easier than using hierarchical measures due to their great variety and necessary parameterizations (cf. Brucker et al. 2011a).

In this work we follow the generalizing strategy of ignoring possible structuring relations over labels. However, in Chapter 7 we introduce an algorithm with an artificial hierarchical problem decomposition and model organization, which may be perfectly used in a HMLC scenario.

## 2.6 Label Dependencies

From a probabilistic point of view, one of the main differences between multilabel and binary or multiclass classification are the possible dependencies in the label output space. In binary and multiclass problems the only observable probabilistic dependence is between the input variables, i.e. the attributes  $x_j$ , and the label variables  $\lambda_i$ . The dependence directly results from the mapping function in Eq. 2.3. A learning algorithm tries to learn exactly this dependence in form of a classifier function  $h$ . In fact, if a classifier provides a score or confidence for its prediction  $\hat{P}$ , this is often regarded as an approximation of  $P(P = \hat{P} \mid \mathbf{x})$ , i.e. the probability that  $\hat{P}$  is true given a document  $\mathbf{x}$ .

As mentioned above, we may additionally observe dependencies between labels in multilabel classification. I.e. we may observe that the occurrence or absence of single labels under certain circumstances *correlate* with each other. From the early beginning of MLC, there have been attempts to exploit these types of *label correlations* (cf. Section 2.8, 8.1). However, only recently Dembczyński et al. (2010b) provided a clarification and formalization of label dependence in multilabel classifications. Following their argumentation, one must distinguish between unconditional and conditional label dependence. Roughly speaking, *unconditional dependence* or independence of labels does not depend on a specific given input instance (the condition) while *conditional dependence* does. An example may illustrate this.

Suppose a label space indicating topics from news articles, and suppose further that  $\lambda_u$  is the topic *politics* and  $\lambda_v$  corresponds to *foreign affairs*. Especially if the topics are orga-



nized in a hierarchy and it holds that  $\lambda_u \succ \lambda_v$  (cf. Section 2.5.6), there will obviously be a dependency between both labels. We will hence observe  $y_u$  with a different probability  $P(y_u = 1) < 1$  when  $y_v$  is also observed, since  $P(y_u = 1|y_v = 1) = 1$ . The probability  $P(y_v = 1|y_u = 1)$  of seeing an article about *foreign affairs* on a page in the politics section will in turn be also much higher than by just randomly opening the newspaper, which corresponds to  $P(y_v = 1)$ . These probabilities are *unconditional* since they do not depend on a particular document. Suppose now that a news article is about the *Euro crisis*. The *conditional* probabilities  $P(\lambda_u = 1|\mathbf{x})$ ,  $P(\lambda_v = 1|\mathbf{x})$  and  $P(y_v = 1|y_u = 1, \mathbf{x})$  would likely increase and hence be different from the unconditional ones. However, if an article was about the *cardiovascular problems of Ötzi*, we would observe that both labels are *conditionally independent*, since (very likely)  $P(y_u = a|y_v = b, \mathbf{x}) = P(y_u|\mathbf{x}) = 1 - a$  for all  $a, b \in \{0, 1\}$  and interchanged  $u$  and  $v$ .

Formally,<sup>14</sup> we define the joint probability for a label vector as  $P(\mathbf{y})$  and compute the marginal distribution of a label as

$$P(y_i = b|\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}, y_i = b} P(\mathbf{y}|\mathbf{x}) \quad (2.27)$$

for  $b \in \{0, 1\}$ . Note that in contrast to multiclass and binary classification, it does not hold that  $\sum_i P(y_i = 1) = 1$  but instead  $\sum_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y}) = 1$  due to the different output spaces. According to the product rule of probability, the joint distribution can be written as

$$P(\mathbf{y}|\mathbf{x}) = P(y_1|\mathbf{x}) \cdot P(y_2|y_1, \mathbf{x}) \cdot \dots = P(y_1|\mathbf{x}) \prod_{i=2}^n P(y_i|y_1, \dots, y_{i-1}, \mathbf{x}) \quad (2.28)$$

Hence, a vector of labels  $\mathbf{y}$  is called *conditionally (in)dependent* or *unconditionally (in)dependent*, respectively, if

$$P(y_1|\mathbf{x}) \prod_{i=2}^n P(y_i|y_1, \dots, y_{i-1}, \mathbf{x}) \stackrel{(\neq)}{=} \prod_{i=1}^n P(y_i|\mathbf{x}) \quad (\text{conditionally}) \quad (2.29)$$

$$P(y_1) \prod_{i=2}^n P(y_i|y_1, \dots, y_{i-1}) \stackrel{(\neq)}{=} \prod_{i=1}^n P(y_i) \quad (\text{unconditionally}) \quad (2.30)$$

Note that neither dependence implies the other.

Similarly, we can define the dependence between only a subset of labels. The degree of dependencies is categorized by Zhang and Zhang (2010) into relations of first, second, and high-order according to the number of labels involved. This division is used to categorize multilabel learners in response to the degree of labels correlations they consider or are able to model and detect. Hence, first-order learners correspond to classifiers

<sup>14</sup> We follow the explanation of Dembczyński et al. (2010b), but will loosen and simplify the argumentation and notation in order to just present the main intuitive idea. In particular, we shall not make a (clear) distinction between realization and corresponding random variable or vector.



which just try to estimate the marginals independently of other labels, while second-order approaches are able to consider pairwise dependencies of the form  $P(y_i|y_j) \neq P(y_i)$ , whereas it is not said per se whether conditional or unconditional, and high-order learners go beyond. Some learners which explicitly respect correlations are reviewed in Section 8.1, 2.8 and 3.5.7.

## 2.7 Evaluation Measures of Predictive Quality

There is no generally accepted procedure for evaluating multilabel classifications. Therefore, several measures from multiclass classification and from information retrieval were adopted and adapted in order to measure multilabel effectivity. Depending on the type of prediction a evaluation function considers, we distinguish between two types of metrics: bipartition and ranking evaluation measures. The first type of error functions  $\delta : 2^{\mathcal{L}} \times 2^{\mathcal{L}} \rightarrow [0; \infty)$  (in a different representation using label vectors,  $\delta : \mathbf{y} \times \mathbf{y} \rightarrow [0; \infty)$ , cf. Eq. 2.8) receives the correct labelset  $P$  and the predicted one  $\hat{P}$ . Accordingly we define the predicted negative labelset as  $\hat{N}$ . The second type of measures compares the predicted ranking  $\hat{\mathbf{r}}$  with the given true labelset:  $P: \delta : \mathcal{L} \times \Pi^n$ .

There exists a wide range of different metrics for MLC, and each measure focuses on and evaluates a different aspects of the classifier. This hence allows to evaluate a classifier according to the specific needs of the application setting or the user, but on the other hand this makes it difficult to evaluate and compare the, roughly speaking, *general performance* of learning algorithms, though certainly the perfect, and hence the best generally performing, classifier is clearly defined (the one which always predicts the true labelset). This aspect will be further discussed in Section 2.7.5.

Firstly, we will review the most popular bipartition or labelset comparing measures in Section 2.7.3. Secondly, the loss functions given in Section 2.7.4 allow to evaluate classifiers that only or in addition return rankings.

### 2.7.1 Aggregation and Averaging

Most of the metrics analyzed are based on evaluations per examples or *example-pivoted* measures, i.e. a metric is computed for each text example, and not for the global prediction on a whole test set. This is reasonable, since the classifiers analyzed produce predictions  $h(\mathbf{x})$  based on an individual and independent example  $\mathbf{x}$ . However, there are settings where global predictions may be desirable, particularly e.g. in object ranking (given a label), but also in classical MLC. Especially bipartition measures are often *label-based*, while this is rather uncommon for ranking losses (for exceptions see e.g. Rubin et al. 2011). Therefore, this work only considers label-based bipartition measures, which are introduced in Section 2.7.3.

We first introduce two aggregations operators which we will need for the formal definitions of measures and different averaging forms. Let  $C_i$  be a sequence of addable and

scalable results, such as (confusion) matrices or simple scalars,  $i = 1, \dots, n$ , then we define the following aggregation operators:

$$\sum_{i=1}^n C_i := C_1 \oplus \dots \oplus C_n \quad (2.31)$$

$$\text{avg}_{i=1}^n C_i := \frac{1}{n} \sum_{i=1}^n C_i \quad (2.32)$$

where  $\oplus$  denotes the cell-wise addition  $A \oplus B = (a_{ij} + b_{ij})_{ij}$  in case of matrices and  $A \oplus B = A + B$  in case of scalars. The definition of the sum might seem superfluous and trivial, but this general formulation allows a simple extension to other types of results such as rankings.

From multiclass classification, information retrieval and many other fields, there exist two well-known averaging strategies in order to obtain a single comparable value on a test set, namely *micro-* and *macro-averaging*. Mainly, the difference lies in the order of the aggregation and the application of the evaluation function: micro-averaging first aggregates, i.e. adds up, the results, which are then evaluated (once) with the error function, while macro-averaged values result from first applying the evaluation function on all available results independently and then aggregating, i.e. building the arithmetic mean, over the computed errors. Hence, micro- corresponds to  $\delta \circ \sum$  and macro-averaging to  $\text{avg} \circ \delta$ , with  $\circ$  denoting the concatenation of operations.

Since there exist two dimensions in MLC over which it is possible to iterate in order to aggregate to one value, namely the labels and test instances, there exist six possible combinations of aggregation, of which only four are mathematically distinct. Let  $i$  iterate over all labels  $\lambda_1 \dots \lambda_n$  and  $j$  over the test examples  $\mathbf{x}_1 \dots \mathbf{x}_{|\mathcal{T}_{est}|}$ , then the four combinations are given by

$$\begin{aligned} \delta \circ \sum_i \circ \sum_j &= \text{(label and example-based) micro-averaged } \delta \\ \delta \circ \sum_j \circ \sum_i &= \text{example-based (macro-)averaged, label-based micro-averaged } \delta \\ \text{avg}_j \circ \delta \circ \sum_i &= \text{label-based (macro-)averaged, example-based micro-averaged } \delta \\ \text{avg}_i \circ \delta \circ \sum_j &= \text{(label and example-based) macro-averaged } \delta \\ \text{avg}_i \circ \text{avg}_j \circ \delta &= \text{(label and example-based) macro-averaged } \delta \\ \text{avg}_j \circ \text{avg}_i \circ \delta &= \text{(label and example-based) macro-averaged } \delta \end{aligned}$$

Most of the bipartition-based measures and all ranking-based measures are example-based averaged, i.e. a metric is computed per example and the resulting test values are averaged with the arithmetic mean operator. Moreover, no computation based on labels

is normally possible for ranking based measures in general. We therefore usually present the final score of an evaluation function on a test set  $\mathcal{T}_{est}$  as

$$\delta_{test}(h, \mathcal{T}_{est}) = \frac{|\mathcal{T}_{rain}| + |\mathcal{T}_{est}|}{\text{avg}_{j=|\mathcal{T}_{rain}|+1}^{|\mathcal{T}_{rain}|+|\mathcal{T}_{est}|}} \delta(P_j, h(P_j)) \quad (2.33)$$

We will refer to scores that were aggregated in this way *example-based averaged* or shorter *example-averaged*.

## 2.7.2 Cross Validation

If cross validation is used, we randomly partition a dataset  $\mathcal{T}$  into  $b$  non-overlapping sets of the same size  $\mathcal{T}^i$ ,  $i = 1 \dots b$  and compute  $b$  scores  $\delta_{test}^i = \delta_{test}(h_{\mathcal{T} \setminus \mathcal{T}^i}, \mathcal{T}^i)$ , which we average in the same way:

$$\delta_{test}^{CV} := \text{avg}_{i=1}^b \delta_{test}^i \quad (2.34)$$

Note that if all buckets  $\mathcal{T}^i$  ideally have the same size, each instance in  $\mathcal{T}$  does receive the same weight in the averaged score. We omit indices of  $\delta$  if the context is clear.

## 2.7.3 Bipartition Evaluation Measures

Our approach is to consider a multilabel classification problem as a meta-classification problem where the task is to separate the set of possible labels into relevant labels and irrelevant labels. Hence, the bipartition evaluation measures are mostly based on already existing measures for multiclass classification, and as such, they are often based on or can always be computed from confusion and contingency matrices.<sup>15</sup> A general, binary confusion matrix is a  $2 \times 2$  dimensional matrix counting the true and false positives, and the true and false negatives of a prediction:

$C$	predicted	not predicted
relevant	$tp$	$fn$
irrelevant	$fp$	$tn$

An atomic confusion matrix, i.e. the matrix with the maximal possible distinction of cases, refers to a specific class  $\lambda_i$  and a specific instance  $\mathbf{x}_j$  and is denoted by  $C_j^i$ . Hence the element  $tp$  ( $tp_i^j$ ) is one if the class was correctly predicted as positive, i.e. if  $y_i = 1 \wedge \hat{y}_i = 1$  for the example  $\mathbf{x}_j$ , and zero otherwise.  $fp$ ,  $tn$  and  $fn$  accordingly cover the cases for which  $y_i = 0 \wedge \hat{y}_i = 1$ ,  $y_i = 0 \wedge \hat{y}_i = 0$  and  $y_i = 1 \wedge \hat{y}_i = 0$  apply.

A bipartition evaluation function is defined in terms of a confusion matrix, i.e.  $\delta : \mathbb{N}^{2 \times 2} \rightarrow [0; \infty)$ . For the evaluation on a test set  $\mathcal{T}_{est}$  we obtain  $m \cdot n$  atomic confusion

<sup>15</sup> Indeed, we make no distinction between the terms *confusion matrix based* and *bipartition* or *labelset* evaluation measure since they denote mathematically exactly the same.

matrices, but since we want to obtain one final (one-dimensional) score, it is necessary to aggregate confusion matrices or scores. Since we are mainly interested in example evaluating measures, we will formulate the evaluation functions if possible in terms of an example-based aggregated confusion matrix, i.e. in terms of the true labelset  $P$ , the predicted one  $\hat{P}$  and the complementary negative labels  $N$  and  $\hat{N}$ :

$C_j = \sum_i C_j^i$	predicted	not predicted	
relevant	$ P_j \cap \hat{P}_j $	$ P_j \cap \hat{N}_j $	$ P_j $
irrelevant	$ N_j \cap \hat{P}_j $	$ N_j \cap \hat{N}_j $	$ N_j $
	$ \hat{P}_j $	$ \hat{N}_j $	$ \mathcal{L} $

From such a sequence of confusion matrices  $C_j$ , we can compute the following well-known measures:

- **Recall, Precision and  $F$ -measure**

*Precision* (PREC) computes the percentage of predicted labels that are relevant, *recall* (REC) computes the percentage of relevant labels that are predicted, and the  $F1$ -measure is the harmonic mean between the two. In terms of  $tp, fp, tn, fn$  of a confusion matrix  $C$ , we obtain the following general definition:

$$\text{PREC}(C) := \frac{tp}{tp + fp} \quad (2.35)$$

$$\text{REC}(C) := \frac{tp}{tp + fn} \quad (2.36)$$

$$\text{F1}(C) := \frac{2}{\frac{1}{\text{REC}(C)} + \frac{1}{\text{PREC}(C)}} = \frac{2 \text{REC}(C) \text{PREC}(C)}{\text{REC}(C) + \text{PREC}(C)} \quad (2.37)$$

But since we focus on example based predictions, we use the following more handy formulation based on the label-based micro-averaged  $C_j$  or  $P$  and  $\hat{P}$ , respectively:

$$\text{PREC}(C_j) = \text{PREC}(P_j, \hat{P}_j) := \frac{|\hat{P}_j \cap P_j|}{|\hat{P}_j|} \quad (2.38)$$

$$\text{REC}(C_j) = \text{REC}(P_j, \hat{P}_j) := \frac{|\hat{P}_j \cap P_j|}{|P_j|} \quad (2.39)$$

$$\text{F1}(C_j) = \text{F1}(P_j, \hat{P}_j) := \frac{2}{\frac{1}{\text{REC}(P_j, \hat{P}_j)} + \frac{1}{\text{PREC}(P_j, \hat{P}_j)}} = \frac{2 \text{REC}(P_j, \hat{P}_j) \text{PREC}(P_j, \hat{P}_j)}{\text{REC}(P_j, \hat{P}_j) + \text{PREC}(P_j, \hat{P}_j)} \quad (2.40)$$

In the literature, we find both example and label-based micro-averaged as well as example-averaged label-based micro-averaged recall, precision and  $F$ -values (e.g.

Lewis et al. 2004, Sebastiani 2002, Yang and Liu 1999). We use the following abbreviations:

$${}_{mm}\text{PREC} = \text{PREC}\left(\sum_{i,j} C_j^i\right) \quad {}_{mm}\text{REC} = \text{REC}\left(\sum_{i,j} C_j^i\right) \quad {}_{mm}\text{F1} = \text{F1}\left(\sum_{i,j} C_j^i\right) \quad (2.41)$$

$${}_{Mm}\text{PREC} = \text{avg}_j \text{PREC}\left(\sum_i C_j^i\right) \quad {}_{Mm}\text{REC} = \text{avg}_j \text{REC}\left(\sum_i C_j^i\right) \quad {}_{Mm}\text{F1} = \text{avg}_j \text{F1}\left(\sum_i C_j^i\right) \quad (2.42)$$

The (macro-)averaging over the examples considers each example with the same weight, in contrast to the  ${}_{mm}\delta$  measures, which take examples with greater associated labelsets more into account than those with small labelsets. Note e.g. that the precision on an example with  $|P| = 1$  only contributes with  $1/(\sum_{j=1}^m |P_j|)$  to the overall precision instead of  $\frac{1}{m}$  in the case of macro-averaging. On the other hand, and independently of whether it may be desirable or not to emphasize correct predictions on examples with a higher number of labels (which could be reasonably considered more difficult to correctly classify), recall and precision are not defined on examples for which  $|P| = 0$  or  $|\hat{P}| = 0$ , respectively. The difficult and often unmotivated decision, whether recall and precision are defined as zero or one for these cases, or the examples are simply ignored, makes it preferable for many authors and many applications (e.g. datasets where this could happen) to use the micro-micro averaged versions.

These complications could also be the reason why the label-based (macro-) averaged, example-based micro-averaged aggregation and the label/example-based macro-averaged aggregation of recall and precision are classically only rarely computed or compared in the literature (cf. e.g. surveys Sebastiani 2002, Tsoumakas and Katakis 2007), though macro-averaging over the labels is known to allow a more commensurate and balanced evaluation of the performance on small labels, i.e. labels with a small number of associated examples. On the other hand, very small labels are often not interesting for the application in focus. Montejo Ráez et al. (2004) e.g. actively eliminate models for labels for which the predictive accuracy is too low, resulting effectively in ignoring small labels. As the authors point out, this approach only slightly hurts recall but should enhance precision. Note also that just as micro-averaging could underestimate small classes, macro-averaging could underrate the effectiveness on big classes, though their high frequency is often precisely a sign for their high importance. This aspect is also emphasized in the discussion of the macro-averaged Hamming loss in the following.

As a convention for simplification, we may omit the micro and macro indicating indices if it is clear from the context and especially if example and label-based micro-averaging is being used.

- **Hamming Loss**

The *Hamming loss* (HAMLOSS) computes the percentage of labels that are misclassified, i.e., relevant labels that are not predicted or irrelevant labels that are predicted. In terms of a single example  $\mathbf{x}_j$ , the Hamming loss is defined as

$$\text{HAMLOSS}(\hat{P}_j, P_j) := \frac{1}{|\mathcal{L}|} |\hat{P}_j \Delta P_j| \quad (2.43)$$

The operator  $\Delta$  denotes the symmetric difference between two sets and is defined as  $A \Delta B := (A \setminus B) \cup (B \setminus A)$ , i.e.  $\hat{P}_x \Delta P_x$  has all labels that only appear in one of the two sets.

Hamming loss basically corresponds to the macro-averaged classification error in the confusion matrix, i.e.  $\text{HAMLOSS}(C_j) = \text{avg}_i I[y_i \neq \hat{y}_i]$ , as was already pointed out by [Tsoumakas and Vlahavas \(2007\)](#). In terms of the confusion matrix, in we write it as

$$\text{HAMLOSS}(C_j) := \frac{fp + fn}{tp + fp + tn + fn} = \frac{fp + fn}{n} \quad (2.44)$$

It is therefore also very popular in the literature, but unfortunately Hamming loss generally favors algorithms with high precision and low recall. The following intuitive argumentation illustrates this.

Since the average labelset size  $\text{avg}_j |P_j|$  is usually much smaller than  $n$ , predicting the empty labelset is already a good strategy for optimizing Hamming loss. And starting from this, it will hence almost never pay out to predict close to or more than  $\text{avg}_j |P_j|$  labels on average, since the probability rapidly increases that the Hamming loss is deteriorated from changing a prediction  $\hat{y} = 0$  to  $\hat{y} = 1$ . Roughly estimated, an algorithm can only expect to benefit from giving a positive prediction for a label if the probability that this class is relevant is higher than  $1 - \text{avg}_j |P_j|/n$ , which is usually rather high.

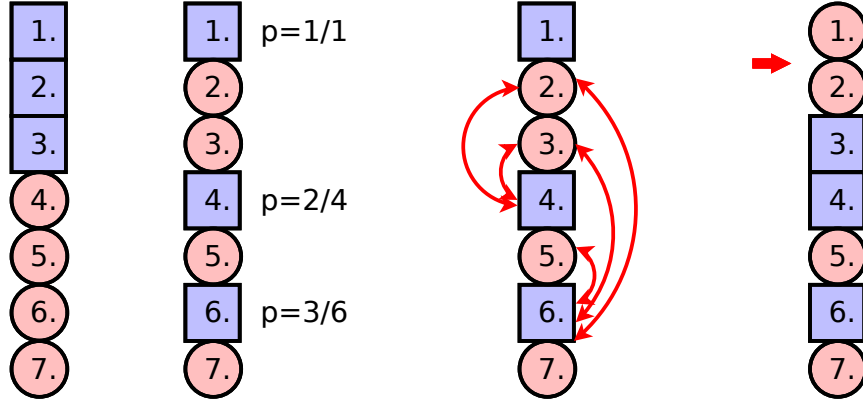
In this particular case, we see that macro-averaging gives too much weight to unfrequent labels and simultaneously underrates the most frequent classes. Hence, recall and precision (or F1) allow a much more commensurate evaluation of a classifier.

- **Subset Accuracy**

The subset accuracy (Acc), sometimes also called classification accuracy, indicates whether a prediction  $\hat{P}$  exactly matches the true labelset  $P$  (cf. e.g. [Ghamrawi and McCallum 2005](#), [Zhu et al. 2005](#)).

$$\text{Acc}(P_i, \hat{P}_i) := I[P_i = \hat{P}_i] \quad (2.45)$$

The example-averaged mean hence denotes the percentage of perfectly predicted labelsets on the test set.



**Figure 2.3:** Diagrams of predicted label rankings and measures. Blue rectangles denote positive classes, red circles negatives. First ranking: perfect classification, all relevant classes are ranked over irrelevant ones, all measures are zero, except from AvgP and  $F1_{|P|} = 1$ . Second and third ranking: classes on position 4 and 6 are misplaced, thus 5 of 12 possible pairs of labels are not correctly ordered, top class is correct,  $RANKLOSS = 5/12$ ,  $MARGIN = 4$ ,  $ONEERR = 0$ ,  $ISERR = 1$ ,  $AVGP = 2/3$ ,  $F1_{|P|} = 1/3$ . Last ranking: top class is wrong,  $ONEERR = 1$ .

Although this metric evaluates exactly the fulfillment of the main objective of an multilabel classifier, namely to correctly predict the true labelset, its strictness makes the measure almost useless for problems with a high number of classes or which are hard to learn. For these problems, the measure becomes almost zero and the differences between different approaches become very small or disappear though differences may be obvious for correct partial prediction.

If not otherwise stated, the example-averaged means of the metrics are given in experimental results.

## 2.7.4 Ranking Quality Measures

In order to evaluate the predicted ranking we use different *ranking losses*. The losses are computed comparing the ranking with the true set of relevant classes, each of them focusing on different aspects. We recall that  $r(\lambda)$  returns the position of label  $\lambda$  in a ranking  $\mathbf{r}$  and that  $r^{-1}(p)$  has as outcome the label at position  $p$ . Since there is no true ranking, only the predicted ranking, we omit the otherwise used distinctive sign in  $\hat{\mathbf{r}}$ . Based on this and given a test example  $\mathbf{x}$ , the different metrics are computed as follows and as illustrated in Figure 2.3.

- **Is-Error Loss**

The *is-error loss* ( $ISERR$ ) determines whether  $r(\lambda) < r(\lambda')$  for all relevant classes  $\lambda \in P$  and all irrelevant classes  $\lambda' \in N$ . It returns 0 for a completely correct, *perfect ranking*, and 1 for an incorrect ranking, irrespective of ‘how wrong’ the ranking is.

With the help of the error set  $\mathcal{E}$  of incorrectly ordered label pairs, we define  $\text{IsERR}$  as follows:

$$\begin{aligned}\mathcal{E} &:= \{(\lambda, \lambda') \mid r(\lambda) > r(\lambda')\} \subseteq P \times N \\ \text{IsERR}(P, \mathbf{r}) &:= I[\mathcal{E} \neq \emptyset]\end{aligned}\tag{2.46}$$

Note that is-error is closely related to subset accuracy on page 30. If a label ranking classifier is used that uses thresholding for predicting a labelset from the ranking,<sup>16</sup> then Is-Error already gives an upper bound of  $1 - \text{IsERR}$  for the obtainable subset accuracy. In other words, is-error evaluates the label ranking effectivity of such a classifier while the gap to subset accuracy indicates the bipartition quality of the used approach.

- **One-Error Loss**

The *one-error loss* ( $\text{ONEERR}$ ) determines whether the top-ranked label is relevant or not, and ignores the relevancy of all other labels.

$$\text{ONEERR}(P, \mathbf{r}) := I[r^{-1}(1) \notin P]\tag{2.47}$$

- **Error Set Size and Ranking Loss**

The *error set size loss* ( $\text{ERRSETSIZE}$ ) computes the number of pairs of labels which are not correctly ordered (Crammer and Singer 2003). As  $\text{IsERR}$ , it is 0 for a perfect ranking, but it additionally differentiates between different degrees of errors.

$$\text{ERRSETSIZE}(P, \mathbf{r}) := |\mathcal{E}|\tag{2.48}$$

The normalized version *ranking loss* ( $\text{RANKLOSS}$ ) is more frequently used and computes the average fraction of erroneously ordered pairs of labels.

$$\text{RANKLOSS}(P, \mathbf{r}) := \frac{|\mathcal{E}|}{|P| \cdot |N|}\tag{2.49}$$

Note that  $\text{ERRSETSIZE}$  determined on a test set cannot be normalized afterwards to  $\text{RANKLOSS}$  since the denominator may change for each test example. Error set size loss gives examples that have more labels associated, and hence could suffer more prediction errors, more weight than ranking loss.

Note also the close connection of ranking loss to other metrics from other fields, such as the Kendall's tau  $\tau$  distance between two rankings, which analogously measures the number of *discordant* pairs. As Kotłowski et al. (2011), Rubin et al. (2011) pointed out, ranking loss is also equivalent to the area ‘over’ the ROC curve (cf. p. 34), i.e.  $\text{RANKLOSS} = 1 - \text{AUC}_{\text{ROC}}$ .

<sup>16</sup> More specifically, some type of ranking cut-off strategy in which the labels are selected from the beginning of the ranking until a certain position but without leaving gaps, see also Section 3.5.4.



- **Margin Loss**

The margin loss (MARGIN) returns the number of positions between the worst ranked positive and the best ranked negative classes, and was first introduced by [Loza Mencía \(2006\)](#) and subsequently used e.g. by [Rubin et al. \(2011\)](#). This metric is directly related to the number of wrongly ranked classes, i.e. the positive classes that are ordered below a negative class, or vice versa. We refer to this set by  $\mathcal{F}$ .

$$\begin{aligned}\mathcal{F} &:= \{\lambda \in P \mid r(\lambda) > r(\lambda'), \lambda' \in N\} \\ &\quad \cup \{\lambda' \in N \mid r(\lambda) > r(\lambda'), \lambda \in P\} \\ \text{MARGIN}(P, \mathbf{r}) &:= \max \left( \max\{r(\lambda) \mid \lambda \in P\} \right. \\ &\quad \left. - \min\{r(\lambda') \mid \lambda' \notin P\}, 0 \right) \\ &= \max(|\mathcal{F}| - 1, 0) = |\mathcal{F}| - \text{IsERR}(P, \mathbf{r})\end{aligned}\tag{2.50}$$

Margin loss is very similar to *Coverage* (cf. e.g. [Tsoumakas et al. 2010](#)), which counts the number of positions in the ranking until all relevant labels are covered.

- **Average Precision**

*Average precision* (AvGP) is commonly used in Information Retrieval and computes for each relevant label the percentage of relevant labels among all labels that are ranked before it, and averages these percentages over all relevant labels.

$$\text{AvGP}(P, \mathbf{r}) := \frac{1}{P} \sum_{\lambda \in P} \frac{|\{\lambda' \in P \mid \mathbf{r}(\lambda') \leq \mathbf{r}(\lambda)\}|}{\mathbf{r}(\lambda)}\tag{2.51}$$

These ranking measures are computed for each example and then averaged over all examples, in contrast to the following metrics.

- **Median Recall, Precision and *F*-measure**

For a label ranker without any explicit bipartition strategy, a reasonable strategy is to always predict the number of labels that can be expected by observing the training set. Therefore, we compute the median over the labelset sizes in the training set and use this value as the position where the rankings are cut off (cf. [Fürnkranz et al. 2008](#)).

Let  $t_k : \Pi_{n,n}^n \rightarrow \mathbb{N}^{2 \times 2}$  be a function that generates the confusion matrix for a cut off at position  $k$ , i.e. the confusion matrix corresponding to the prediction  $\hat{P}_j = \{r^{-1}(1), \dots, r^{-1}(k)\}$ , let  $d$  be the median over all  $|P_j|$  in the training set  $\mathcal{T}_{rain}$ , then

we define the example and label-based micro-averaged *median recall, precision and F1-measures* as:

$${}_{mm}\text{PREC}_d := \text{PREC}\left(\sum_{j=1}^{|\mathcal{T}_{est}|} \theta_d(\mathbf{r}_j)\right) \quad (2.52)$$

$${}_{mm}\text{REC}_d := \text{REC}\left(\sum_{j=1}^{|\mathcal{T}_{est}|} \theta_d(\mathbf{r}_j)\right) \quad (2.53)$$

$${}_{mm}\text{F1}_d := \text{F1}\left(\sum_{j=1}^{|\mathcal{T}_{est}|} \theta_d(\mathbf{r}_j)\right) \quad (2.54)$$

- **Idealistic F1**

For the (micro-averaged) *idealistic* F1 ( $\text{F1}_{|p|}$ ), we compute analogously to the median variants the F1-measure as if exactly the right number of  $|P_j|$  labels was returned for each test example  $\mathbf{x}_j$  (cf. [Fürnkranz et al. 2008](#)):

$${}_{mm}\text{F1}_{|p|} := \text{F1}\left(\sum_{j=1}^{|\mathcal{T}_{est}|} \theta_{|P_j|}(\mathbf{r}_j)\right) \quad (2.55)$$

Since the denominators in Eq. 2.37 coincide,  ${}_{mm}\text{F1}_{|p|}$  is equal to  ${}_{mm}\text{PREC}_{|p|}$  and  ${}_{mm}\text{REC}_{|p|}$ . This equivalence could suggest to determine the metric as break-even point of precision and recall. But note that the performance that we called “idealistic” is not necessarily the optimal choice in the sense that it is the highest achievable value. Higher F1-values are achievable if we deviate from the original number of labels, because of a suboptimal ranking of the labels. The optimal boundary for each example depends on both the example and the predicted ranking.

- **ROC and Recall/Precision Curves**

In order to visualize graphically the behavior over all possible boundaries, we can compute the receiver operating characteristic (ROC) and recall/precision curves. This is done by building a confusion matrix  $\theta_b(\mathbf{r}_j)$  for each possible boundary  $b = 0 \dots n$  (i.e., on all positions in the ranking) and test example  $\mathbf{x}_j$ . The respective recall/precision and true positive/false positive values are then plotted resulting in a polygon with  $n$  segments. We use micro-averaging to average these curves by computing summary matrices

$$\sum_{j=1}^{|\mathcal{T}_{est}|} \theta_b(\mathbf{r}_j) \quad (2.56)$$

for  $b = 0 \dots n$ . Recall/precision and ROC curves are plotted using the recall/precision and  $tp / fp$  values from these confusion matrices.

---

### 2.7.5 Discussion

This section tried to provide a comprehensive overview of the landscape of multilabel metrics. However, though several measures were presented, we focused on those that appear in the experimental section of this thesis and hence only a subset of the existing and employed metrics in the literature were gathered. The list of hierarchical measures is even more extensive (cf. Brucker et al. 2011a).

In the following, we try to group the presented measures in order to allow a reasonable and balanced selection at time of evaluation. We focus on possible equivalences and general objectives.

- $\text{PREC}$ ,  $\text{REC}$ ,  $\text{F1}$  and  $\text{AvGP}$  reward predictions with a high density of relevant labels at the top. Particularly precision and recall allow a fine grained analysis of bipartition ability of an algorithm. The interesting macro-average performance is often omitted in the evaluations, mainly due to the mentioned complications in the computation and of course also because it is mostly much lower than the respective micro-averaged measures (which should obviously not be a criterion).  $\text{AvGP}$  and the idealistic variant of  $\text{F1}$  allow to evaluate rankings, though  $\text{F1}_{|\mathcal{P}|}$  additionally serves as a bound for micro-averaged  $\text{F1}$ . Note that  $\text{AvGP}$  does often not correlate to  $\text{RANKLOSS}$  or similar measures particularly in the magnitude of the differences due to the different objectives.
- $\text{RANKLOSS}$ ,  $\text{MARGIN}$ , coverage, area under the ROC curve ( $\text{AUC}$ ) measure the ranking performance. Though  $\text{MARGIN}$  and coverage do not measure errors in the pairings,  $\text{RANKLOSS}$  is bounded by these too. Hence, if the interpretability of  $\text{MARGIN}$  is not needed, it is sufficient to concentrate on  $\text{RANKLOSS}$  (or the reverse  $\text{AUC}$ ).
- $\text{ACC}$  and  $\text{ISERR}$  both indicate the ratio of perfect predictions for ranking and bipartitioning. Therefore,  $\text{ACC}$  is bounded by  $1 - \text{ISERR}$ . In combination they allow hence a good evaluation of the bipartitioning capabilities of an algorithm.<sup>17</sup>
- $\text{HAMLOSS}$  and  $\text{ONEERR}$  share the same property of limited expressiveness particularly for a large number of labels. Hamming loss could be replaced by the Jaccard distance  $1 - |P \cap \hat{P}| / |P \cup \hat{P}| = (fp + fn) / (tp + fp + fn)$ , which only measures the error on labels that are set or were predicted and hence alleviates the problem of the large number of true negatives.

Some of the measures were analyzed by Sokolova and Guy (2009) in view of invariance and their ability to detect changes in the confusion matrix.

In the sum, the author believes that micro-averaged recall, precision for bipartitioning approaches and in addition  $\text{RANKLOSS}$ ,  $\text{AvGP}$  and the potential  $\text{F1}_{|\mathcal{P}|}$  for ranking allows a commensurate, general evaluation of a multilabel classifier. If the number of classes is low, it is further advisable to measure  $\text{ACC}$  and  $\text{ISERR}$ .

---

<sup>17</sup> Another interesting point is that one can deduce the example-based mean of any other measure on the mistakes in the test set using the overall mean by computing  $\delta / \text{ISERR}$  or  $\delta / (1 - \text{ACC})$  respectively.

---

However, this always depends on the particular application. Moreover, as Dembczyński et al. (2010c) pointed out, some of the measures, e.g. subset accuracy on the one side and hamming loss on the other, are not compatible in the sense that optimizing one does not necessarily optimize the other.<sup>18</sup> This makes it also clear that the objective should be clarified before the selection or development of a learner. Hence, statements about a general or average good performance of a new method with respect to a set of different measures without differentiating and without interpretation should be taken with care. A purposeful analysis should be preferred.

## 2.8 Multilabel Learning Algorithms

In a survey on multilabel classification, Tsoumakas and Katakis (2007) distinguish between two categories of approaches, namely, those which employ a *transformation* into a set of binary classification problems and approaches *adapting* existing methods to handle multilabeled data directly. As it turns out, it is sometimes difficult to separate both, since many adapted approaches consist in transforming the original problem into a series of easier problems. On the other hand, a series of learners have been developed in the meanwhile that are conceptually at least not directly based on existing approaches but were specifically developed with the multilabel setting in mind.

Thus, we use the following (not exhaustive) coarse division in this work: *transformational* and *holistic* approaches. The first group corresponds to the notion of transformation and decomposition techniques and this work is specifically dedicated to one of the possible techniques. The following section provides a basic description. A profound explanation and analysis is provided in Chapter 3. The latter group of learners try to solve the global problem directly without transforming it into subproblem(s). An overview of existing techniques is given in Section 2.8.2.

As described above, it is sometimes difficult to draw the boundary and some of the developments described in the following may not fit this categorization perfectly well. The sections after Section 2.8.2 review some of the main, older and newer, techniques encountered in multilabel learning. Since many of the cited works combine multiple techniques, some of them could indeed be categorized into several classes – a multilabel categorization. We refer to the discussion sections in the corresponding chapters of this thesis for further relevant and interesting literature. Particularly Section 2.5.6 for hierarchical learners, 3.5.4 for generally applicable thresholding techniques, 6.5 for particularly scalable methods, 8.1 for approaches dedicated to label correlations, 9.7 for stacking, 3.5.6 and 5.5 for pairwise approaches and Section 3.5.7 for a comparison of these with one-against-all decomposition. A further source of information are the excel-

---

<sup>18</sup> Note however that the optimal prediction is always defined for all metrics, namely predicting exactly the correct labelset. But a Hamming loss optimized learner will approach the optimum in a different way than an accuracy optimized one. Graphically, on a graph with HAMLoss on the one axis and Acc on the other, one type of learner would approach the optimum in the upper left corner from below the diagonal and the other from above, possibly in form of a curve.

---

lent surveys of Tsoumakas and Katakis (2007) and Tsoumakas et al. (2010) and, for the particularly interested reader, the respective related work sections in the cited literature.

### 2.8.1 Transformational Approaches

We focus on three basic transformation schemes, namely binary relevance, label powerset and pairwise decomposition. The main idea of these methods is to transform the problem into non-multilabel tasks, particularly binary tasks (except for label powerset, which requires one further step), which can then be solved separately with existing binary learners. Section 4.1 e.g. revises the basics of linear learners such as support vector machines (SVM) and perceptrons. Other learners used in this work include C4.5 decision trees and Naive Bayes. We refer to (Mitchell 1997, Witten and Frank 2005) for explanations of these and other approaches. In the case of transformation into several subtasks, the approach is denoted to be decompositive.

In the binary relevance (BR) or one-against-all (OAA) method, a multilabel problem with  $n$  possible classes is decomposed into  $n$  binary problems. For each subproblem, a binary classifier is trained to predict the relevance of the corresponding class. In the pairwise decomposition approach, one classifier is trained for each pair of classes, i.e., the original problem is decomposed into  $\frac{n(n-1)}{2}$  smaller subproblems. The binary classifiers are trained on examples with a clear preference for one of the two classes. During classification, each base classifier is queried and the prediction is interpreted as a vote for one of its two classes. In the label powerset approach (LP), a meta multiclass problem is constructed where each appearing label combination  $P_i$  is interpreted as one separate class. The meta problem is then solved with a normal multiclass algorithm or with the previously presented decomposition methods. More thorough explanations and analyses are given in the next chapter.

### 2.8.2 Holistic Approaches

We consider *holistic approaches* to be approaches that try to solve a multilabel problem globally and jointly. This often involves solving one single optimization problem to find the decision function(s). Therefore, this strategy is called *single-machine* by Rifkin and Klautau (2004) or *all-at-once* by Rueda et al. (2010). Holistic approaches are often able to learn risk minimizing models with respect to a particular metric. Rifkin and Klautau compares solving  $n$  independent optimization problems, each finding a function  $h_i$  for class  $\lambda_i$ , to solving one global problem containing the  $n$  functions  $h_i$ . The findings indicate that the local approach is more advantageous for the particular case of using SVMs on multiclass problems, although this is not directly transferable to the multilabel case (cf. Section 3.5.7).

The work of Elisseeff and Weston (2001) on adapting the SVM algorithm to the multilabel case is probably one of the most frequently cited works in MLC. A global optimization problem is formulated in order to minimize the ranking loss (RANKLOSS), i.e. that no

irrelevant labels are ranked above relevant ones (Rank-SVM). The hypothesis space consists of  $n$  hyperplanes in the input space (cf. Section 4.1.1), just as it would be the case for a BR ensemble of SVMs. BP-MLL is a popular neural networks algorithm for multilabel data which is trained in order to minimize the number of incorrectly paired labels in the output ranking (Zhang and Zhou 2006). To this end, RANKLOSS is reformulated as a differentiable version, which is a prerequisite for being used in back propagation. BP-MLL is similar to MMP, but more than one hidden layer is used, similarly to MLPP (cf. Chapter 4 and Section 4.5.1). The multiclass multilabel perceptron algorithm (MMP) (Crammer and Singer 2002, 2003) learns similarly to Rank-SVM one prototype for each label by globally optimizing an arbitrary ranking loss (cf. Section 2.7.4). But opposed to Rank-SVM, this is done incrementally for each training example (cf. Section 4.3).

For the general case, we can also consider so called case-based or lazy approaches and rule-based learners such as decision trees as holistic methods, as long as they are not based on binary models.

### Case-based

The  $k$ -nearest neighbor approach (cf. Witten and Frank 2005, Sec. 3.8) to multilabel classification (ML-kNN) is inspired by Bayesian reasoning (Zhang and Zhou 2007). It combines the label distributions of the  $k$  neighbors and the a priori distribution in order to make a prediction. The concept that is followed in (Brinker and Hüllermeier 2006) is closely related to the calibration technique described in Section 3.4.5. Each training instance is assumed to be associated with a tri-partite ranking  $P_x \succ_x \lambda_0 \succ_x N_x$ . The rankings of the  $k$  nearest neighbors are aggregated to one predicted ranking by computing the average ranks of the labels among the  $k$  tri-partite rankings.

### Rule-based learner

An associate multilabel rule learner with several possible labels in the head of the rules was developed by Thabtah et al. (2006). These labels are found in the whole training set, while the multilabel lazy associative approach of Veloso et al. (2007) generates the rules from the neighborhood of a test instance during prediction. Hoeffding trees were adapted for learning large multilabel data streams by Read et al. (2010). These trees are incrementally trainable and it is proven that they approximate non-incremental decision trees trained with infinite training examples. Unfortunately, they are not suitable for streams with concept drifts. Both Zhang et al. (2010b) and Kong and Yu (2011b) propose to use ensembles of random decision trees (RDTs) for multilabel classification. The main idea of these trees is described in more detail in Section 4.6.7.

## 2.8.3 Generative Approaches

The early work of McCallum (1999) constructs generative models for labelsets, which consist of a mixture of topic based word distributions. Parametric mixture models consist of probabilistic generative models for each label, in form of class prototypes (Ueda and Saito 2002). However, the different labelsets are modeled on top with respect to the

class prototypes. The greedy approach successively adds one label to the currently most probable labelset until it finds the maximum. The approach of [Streich and Buhmann \(2008\)](#) assumes that documents are drawn from superpositions of  $n$  distributions  $\mathcal{X}^i$ , one for each label. Unlikely label combinations are discarded during the prediction process in order to reduce the number of comparisons  $2^n$  needed by the brute force approach. Latent Dirichlet allocation (LDA) is an unsupervised approach usually applied in order to generalize whole document corpora which assumes that a document is sampled from a mixture of word distributions, the (virtual) topic models. [Rubin et al. \(2011\)](#) estimate these word distributions directly on the training data adopting the labels as topics. An additional LDA process on top of the labels is applied in order to model dependencies between the labels. More details on this approach can be found in Section 6.5.

#### 2.8.4 Ensembles

One of the first multilabel ensemble techniques was boosting: The text and speech classification system Boostexter ([Schapire and Singer 2000](#)) is based on the multilabel extensions of AdaBoost described by [Schapire and Singer \(1999\)](#). Rakel builds an ensemble of  $m$  label powerset classifiers ([Tsoumakas et al. 2011a](#)). Each LP learner focuses on a randomly chosen subset  $\mathcal{L}_i \subset \mathcal{L}$  with  $k = |\mathcal{L}_i|$ , i.e. the  $i$ -th learner receives  $P_{\mathbf{x}} \cap \mathcal{L}_i$  as training signal for an instance  $\mathbf{x}$ . Rakel is hence potentially able to model label dependencies from 1 to  $k$ -order breaking the limitation of pure LP, which is only able to detect simultaneous absences or presences of label combinations  $\{P_i\}$  given in the training set. Interestingly, the  $k$  parameter can be seen as a fader between BR and LP, from fully independent to fully connected classes. [Read et al. \(2008\)](#) follow a very similar approach with their ensembles of pruned sets (PS). PS uses the LP transformation but prunes away infrequent labelsets  $P$  or decomposes them into more frequent subsets  $P' \subset P$ . The ensemble is created by applying PS on random subsets of the training data and these models seem to outperform Rakel ensembles, which used similar training time. Ensemble variants of probabilistic and simple classifier chains ([Dembczyński et al. 2010a](#), [Read et al. 2009, 2011](#)) ensure a certain robustness towards different sequences of the models. [Read et al.](#) further analyze the combination with bagging, i.e. using random subsets of training instances and features. He finds that around 40% of the instances and features are sufficient in order to obtain comparative accuracy. This also holds for ensembles of BR classifiers and would likely also apply to pairwise ensembles. See Section 3.5.7 for a discussion on general classifier chains.

The following authors use a neural network alignment of the model, but essentially use ensemble techniques in order to determine the hidden layer. [Zhang \(2009\)](#) builds a (non-linear) neural network with one hidden layer (ML-RBF). The neurons in this layer are simply a predetermined number of  $k$  centroids of each class, computed via  $k$ -means clustering (cf. [Witten and Frank 2005](#), Sec. 4.7). They achieve consistently better results than another neural network (BP-MLL, see above), boosting approaches and Rank-SVM on *scene*, *yeast* and *yahoo*. The very recent LIFT algorithm similarly selects centroids (by  $k$ -means) in the positive and negative examples of each one-against-all problem and



then replaces the original features of an instance by the distances to these representatives, separately for each BR subproblem (Zhang 2011). Shi et al. (2011) trains an ensemble of ML-RBF networks and shuffles the prototypes in the hidden layers using an evolutionary operator by optimizing simultaneously accuracy and diversity of the predictions. They outperform the baseline, ensembles of classifier chains and Rakel (see above) on *yeast*, *scene* and *yahoo*.

### 2.8.5 Instance Input Space Transformations

Several approaches rely on enriching or even replacing the input instance attributes by new features that are expected to provide additional information. The hope is often to be able to encode characteristics that help to exploit (conditional or unconditional) label dependencies. An example is the approach in Chapter 9, which adds features indicating the presence of exceptional local feature-labelset combinations. Stacking, i.e. the use of classifier predictions as features for the main learner at the bottom, is another popular approach in this context and several techniques will be revised in more detail in Section 9.7. A special group of stacking approaches organize their learners in chains so that the prediction for a particular class depends on the predictions for the previous classes. Classifier chains (CC, Read et al. 2011) are more exhaustively discussed in Section 3.5.7. Probabilistic CC generalize this concept by using probabilistic base classifiers (Dembczyński et al. 2010a). Their approach ensures a Bayes optimal decision according to the conditional label dependencies, since Eq. 2.28 is indeed approximated by  $P(\mathbf{y}|\mathbf{x}) \approx h'(\mathbf{x}) \prod_{i=2}^n h'_i(\mathbf{x}, y_1, \dots, y_{i-1})$  in their setting.

Some kernel based approaches are presented in the following: Kazawa et al. (2005) adds label specific features to the input-space and uses a specialized kernel in order to measure the label vector similarities. During prediction, a test example is subsequently enriched with the features for all possible label combinations and the highest scoring labelset is predicted. The authors derive an algorithm which is more efficient than this brute-force approach, but still requires  $\mathcal{O}(nm^3)$ . Similarly, the general framework of SVMs for structured output spaces also supports the MLC setting by defining an appropriate kernel (Tsochantaridis et al. 2005). Here, too, a heuristic is necessary in order to circumvent the prohibitively high costs of enumerating all possible outputs.

### 2.8.6 Label Output Space Transformations

Transforming the label output space  $\mathcal{Y}$  into a substituting, lower dimensional  $\mathcal{Y}'$  is a relatively new idea. It is based on the observation that the output space is usually very sparse in multilabel problems (cf. Section 2.9.1) and relies on techniques such as principal component analysis, which "compresses" by projecting to lower dimensional spaces. The objective of these approaches is not the improvement of accuracy, but the reduction of computational costs caused by the often observed direct dependence of training and testing time on the number of possible classes. This dependency is one of the leitmotifs



in this work and hence the development of this topic is followed with special interest by the author. Unfortunately, the works observed so far have to use regression algorithms since the reduced space  $\mathbb{R}^{|\mathcal{Y}'|}$  is not binary anymore. Hashing, and particularly semantic hashing such as spectral hashing (Weiss et al. 2008), is a promising solution in order to be used in the (strictly binary) frame of this work.

Compressed sensing (Hsu et al. 2009b) exploits the sparsity of the output space and uses a compressing linear matrix  $A$ , which is filled with Gaussian, Bernoulli or uniformly distributed values. The binary output space is transformed ( $A\mathcal{Y} \subset \mathbb{R}^k$ ) into a  $k = \mathcal{O}(d)$  dimensional continuous space. Reconstruction techniques known from *error correcting output codes* (cf. Section 3.3) map the prediction back into the original label space. Tai and Lin (2010) in contrast interpret the label output space as a  $n$ -dimensional hypercube with the labelsets at the vertices. A singular value decomposition is used in order to determine new principal directions  $e_i$ . The  $e_i$  hence form the projection matrix  $A$  and the new  $\mathbf{y}'$  contain the projections of the original  $\mathbf{y}$  on the  $|\mathcal{Y}'|$  directions  $e_i$ . Again, regression algorithms are then used to learn  $(\mathbf{x}, \mathbf{y}')$ . Bi and Kwok (2011) focus on hierarchical problems. The label output space is preprocessed in order to consider the hierarchical information and then compressed via kernel PCA (kernel dependency estimation) to an alternative space of only 50 dimensions. Their experiments on biological datasets with more than 4000 classes showed that their 50 regressors consistently improved time costs and accuracy compared to several hierarchical learners cited in Section 2.5.6. Rueda et al. (2010) compare linear dimensionality reduction of the output space applied on the whole problem, solving a single optimization problem, with applying it to each of the pairwise subproblems. It is found that pairwise decomposition is more beneficial in terms of accuracy.

### 2.8.7 Alternative Structures and Formulations

Known multilabel problems often result from a simplification of more complex original tasks, such as hierarchical classification (cf. Section 2.5.6). It is often also possible to induce certain structures on multilabel data which may help the learning process. The following approaches have in common that they rely on such an extended (re)formulation.

Sun et al. (2008) consider a multilabel task as a hypergraph with the instances as nodes and  $n$ -ary edges for each label connecting all associated instances. This representation aims at exploiting label correlations by analyzing the hypergraph spectrum. Tsochantaridis and Hofmann (2002) see MLC in the context of collaborative filtering and interpret labels as users and documents as items. The label matrix  $(\mathbf{y}_i)_i$  is hence used in order to compute a probabilistic latent semantic analysis model which encodes the unconditional inter-label dependencies. Feeding a transductive SVM with this additional information outperforms the baseline SVM. Kong and Yu (2011a) similarly uses kernels on the label output space in order to exploit label correlations on multilabel graph data. They claim that different kernels cover different  $k$ -order dependencies and subsequently find that (infinite-order) RBF kernels substantially outperform polynomial kernels of different degrees and the linear kernel. The work of Kong et al. (2011) relies on relational

---

---

multilabel data. Additional features are assumed that connect instances in the training set, e.g. co-authorship for author objects or common directors for movies. These connections are used to define a vicinity of the instances. The feature set for each training example is then extended by aggregating the neighbors' input and also output features in order to capture intra and inter-instance label dependencies. During prediction, a greedy process begins with the original features and subsequently adds features from the expanded neighborhood. This collective approach clearly outperforms classifier chains and BR.

## 2.9 Datasets and Application Scenarios

Multilabel classification problems appear in a wide range of real world situations and applications. We will give in the following some examples organized by different aspects of the scenarios.

The first distinguishing property is the domain of the objects as well as the labels in a MLC problem.

- **Text**

Text is probably one of the oldest domains in which the demand for categorization appeared, particularly multilabel categorization (cf. [Sebastiani 2002](#)). Moreover, data is easily accessible and processable and vastly available. Hence, it was also one of the first research fields for MLC and continues to be the most represented (see also Section 1.1 and particularly Footnote 1).

Documents can be of very different types: news articles, scientific articles, books, emails, law documents, web pages, subtitles, (micro-)blogs, etc. The great variety is also reflected in the possible dimensions of categorizations: genres, topics, authors, keywords, epochs, writing styles, languages ...

- **Multimedia**

We denote by multimedia any kind of image or auditive data such as photographs, films, music titles, audio recordings etc. Many of the possible categorizations for text also apply for multimedia material. In addition we have object and person recognition, optical character recognition, scene categorization, and for audio the recognition of music instruments, emotions, styles, epochs ...

- **Biology**

Biology and bioinformatics is a relatively new field for computerized data processing. For machine learning, possible domains include the classification of genes and proteins according to their functions.

We can usually classify the label domains according to their structure. Typical hierarchical structures come from topic hierarchies, taxonomies and ontologies (cf. Section 2.5.6). Non-hierarchical structures can be found in the indexing of documents, e.g. the assignment of keywords or keyphrases to scientific articles. Flat hierarchical structures have

---

gained increasing attention with the emergence of the participatory world wide web (cf. e.g. Katakis et al. 2008, Tsoumakas et al. 2008). In this context, keyword indexing is often called collaborative or social tagging and the resulting structure is called a folksonomy (for definitions cf. e.g. Peters 2009). The term *tagging* shall indicate that the vocabulary used is not controlled.

As in multiclass classification, MLC algorithms can be used in applications in which full automation is desired as well as for supportive usage. Examples are bookmarking, email filtering or even news categorization (cf. Chapter 4), in contrast to the suggestion of keywords from a huge set of alternatives in the Web 2.0 (see also Chapter 6), possibly presented as a ranked list (compare to label ranking in Section 2.5.4). The latter scenario may demand for semi-supervised approaches as well as for incremental multilabel learners for the processing of masses or streams of data (cf. Chapter 4).

Related applications to which MLC may be applied are information extraction (e.g. ontology based information extraction or syntactic parsing, cf. Chapter 10), and multi-target, multi-task classification or multi-variate regression, where several target variables have to be predicted simultaneously (cf. Chapter 8). Extended applications take the degrees of assignments into account, e.g. popularly assignments of 0 to 5 for rating action, fun, romance etc. of movies in TV guides, which we may call ordinal or graded multilabel classification (Cheng et al. 2010).

Many of the introduced scenarios are covered by the multilabel datasets presented in the next section.

### 2.9.1 Benchmark Datasets

The datasets that are included in the experimental setups throughout this work cover the main three application areas in which multilabeled data is frequently observed: *text categorization* (8 datasets), *multimedia classification* (6 collections) and *bioinformatics* (two benchmarks). Table 2.2 summarizes the main properties, which are the following:

- **Dimensionality of the output space**

The total number of labels  $n$  is the main characteristic of a (single- or multilabel) multiclass task (cf. Section 2.5). It is a crucial factor for determining the complexity of a dataset. The scalability of a learner often decisively depends on  $n$ . Therefore we find Table 2.2 sorted by this property.

- **Cardinality of the output space**

The average size  $d = \text{avg}_x |P_x|$  of the labelsets in a dataset distinguishes multiclass from multilabel datasets. For multiclass problems the size is one, and also for multilabel problems the value is rather small, in relative as well as absolute terms. This means that usually  $d \ll n$  and  $d < k$  with usually single-digit  $k$ .

The average labelset size often has an impact on the computational costs of a learner, e.g. for pairwise decomposing (cf. Section 3.4) or generative (cf. Section 2.8.3) approaches.

**Table 2.2:** Statistics of multilabel datasets used in this work ordered according to the number of classes. The attribute number indicates the original number without any feature subset selection or expansion. *Labelset size*  $d$  denotes the average number of labels per instance, and *label density* indicates the average number of labels per instance  $d$  relative to the total number of classes  $n$ .

dataset name	domain	#instances $m$	#attributes $a$	#labels $n$	labelset size $d$	density $\frac{d}{n}$	distinct $ \{P_x\} $
scene	image	2407	294	6	1.074	17.9 %	15
emotions	music	593	72	6	1.869	31.1 %	27
yeast	biology	2417	103	14	4.237	30.3 %	198
tmc2007	text	28596	49060	22	2.158	9.8 %	1341
genbase	biology	662	1186	27	1.252	4.6 %	32
medical	text	978	1449	45	1.245	2.8 %	94
enron	text	1702	1001	53	3.378	6.4 %	753
mediamill	video	43907	120	101	4.376	4.3 %	6555
rcv1	text	804414	231188	101	3.241	3.1 %	13922
r21578	text	11367	21474	120	1.258	1.0 %	533
jmlr2003	image	65362	46	153	3.071	2.0 %	3115
bibtex	text	7395	1836	159	2.402	1.5 %	2856
eccv2002	image	47065	36	374	3.525	0.9 %	3175
hifind	music	32971	98	623	37.304	6.0 %	32734
delicious	text	16105	500	983	19.020	1.9 %	15806
EUR-Lex	text	19348	166448				
subject matter	"	"	"	201	2.213	1.1 %	2504
directory code	"	"	"	410	1.292	0.3 %	1615
EUROVOC	"	"	"	3956	5.317	0.1 %	16467

- **Label Density**

The density denotes the relative cardinality of a multilabel problem, i.e. the proportion of labels that are on average relevant for an instance in the dataset. The sparsity of the output label matrix is usually very high for multilabel problems, especially for high dimensional datasets with respect to the total number of labels.

- **Diversity of the output space**

The number of distinct labelsets  $P_x$  in a dataset is an indicator for the dependencies between labels. The smaller this value, the fewer patterns in  $\mathcal{Y}$  were used and the more correlated the labels are expected to be. A rough estimation of the upper bound shall be  $(1 + d/n)^n \leq 2^n$ .

Some of the points correspond to the general challenges presented in Section 1.1, as do the following properties, which are not specific to multilabel datasets:

- **Number of instances**

Fully classified data is usually sparse, since manual annotation by experts is very time-consuming and expensive. With the recent emergence of new technologies on

---

the world wide web and the massive participation of normal users in the organization of data, the availability of data has substantially increased. Nevertheless the number of (training) instances continues being a crucial time factor in classifier training.

- **Dimensionality of the input space**

The number of attributes (cf. Section 2.1) is particularly high for text classification problems. This circumstance often requires a feature subset selection since many learning algorithms are very susceptible to this factor. On the other hand, the input vectors are often very sparse (the density of the input matrix is not reflected in Table 2.2), which can be exploited especially by learners working directly in the vector space (cf. Section 4.5.2.4).

Further details, such as feature selection and train/test-splits employed, are given in the sections describing the particular experiments if necessary.

### 2.9.1.1 Text

For the first three text corpora presented it was necessary to do text preprocessing. The last one was even collected, processed and constructed entirely by the author. The remaining corpora were directly used in the form they were published in the respective repositories (cf. Section 2.9.2).

The *Reuters Corpus Volume I* (*rcv1*) is one of the most widely used test collection for text categorization research. It contains 804,414 newswire documents. In general, we use a version which was split into 535,987 training documents (all documents before and including April 26th, 1999) and 268,427 test documents (all documents after April 26th, 1999) for experimentation. We used the token files of Lewis et al. (2004) called *RCV1-v2/LYRL2004*, which are already word-stemmed and stop word reduced. However we repeated the stop word reduction as we experienced that there were still a few occurrences. The 25,000 most frequent features on the training set were selected and weighted with TF-IDF weights (Salton and Buckley 1988). We did not restrict the set of 103 categories although one class does not contain any examples in the training set. More precise details to the preprocessing can be found in Section 6.1 describing *EUR-Lex* or in previous work of the author (Loza Mencía 2006). Details on the distributions of labels, in particular in comparison to *EUR-Lex*, can also be found in Section 6.1.

We also experimented with the older *Reuters 21578* corpus (*r21578*) (Lewis 2004), which has 11,367 examples and 120 possible labels. Through similar preprocessing as in the *rcv1* dataset, we obtain 10,000 features for this dataset.

The *EUR-Lex* is the most challenging dataset in our collection, containing 19,348 legislative documents from the European Union. It was introduced by Loza Mencía and Fürnkranz (2007a) and made publicly available in July 2009 at <http://www.ke.tu-darmstadt.de/resources/eurlex/>. The documents are classified according to three different classification schemes: *subject matter* with 201 classes, *directory code* with 410 classes and *EUROVOC* with 3956 classes, although slightly differing versions were used

---

since the first publication by Loza Mencía and Fürnkranz (2007a). After a similar preprocessing as for *rcv1* and *r21578*, we obtained 5,000 features. Full details of the preprocessing and the characteristics of the dataset are presented in Section 6.1.

Other text classification datasets include *medical* from the Computational Medicine Center’s 2007 Medical Natural Language Processing Challenge that aimed at assigning codes from the International Classification of Diseases to clinical free texts (Read 2012), the *enron* dataset of business-related emails from the Enron Corp. management<sup>19</sup>, *bookmarks* and *bibtex*, collections from the social bookmarking platform BibSonomy (Katakis et al. 2008), the *tmc2007* dataset from the SIAM Text Mining Workshop 2007 of aviation safety reports assigned to flight problem types (Srivastava and Zane-Ulman 2005), and the large *delicious* dataset extracted from the *del.icio.us* social bookmarking platform (Tsoumakas et al. 2008).

### 2.9.1.2 Multimedia

The task in the *scene* dataset (Boutell et al. 2004), one of the most popular datasets in the literature, is to recognize which of six possible scenes (*beach*, *sunset*, *field*, *fall foliage*, *mountain*, *urban*) can be found in a 2407 pictures. Many pictures contain more than one scene. For each image, spatial color moments are used as features. Each picture is divided into 49 blocks using a  $7 \times 7$  grid. A picture is then represented using the mean and variance of each color band of each block, i.e., using a total of  $2 \times 3 \times 7 \times 7 = 294$  features.

The *hifind* collection contains 32,769 music titles annotated on average with 37 from 632 different labels (Pachet and Roy 2009). Details on the acoustic classes are given in Section 8.4.

Duygulu et al. (2002) constructed *eccv2002*, a popular benchmark for image classification and annotation methods, which is also frequently named *corel5k* in a slightly different form (Tsoumakas 2012). It is based on 5000 Corel images, 4500 of which are used for training and the rest 500 for testing.

The collection *jmlr2003* is produced from the first subset (001) of the data accompanying (Barnard et al. 2003). It is based on 6932 images, 5188 of which are used to create the training set and the remaining 1744 to create the test set. The dataset is also named *corel16k* in the research community (Tsoumakas 2012).

The *mediamill* benchmark is based on the Mediamill Challenge dataset (Snoek et al. 2006). It contains pre-computed low-level multimedia features from the 85 hours of international broadcast news video of the TRECVID 2005/2006 benchmark.

In *emotions*, the task is to assign emotions to music (Trohidis et al. 2008). The collection consists of 593 songs selected from 233 music albums and represented by 72 auditive timbre features extracted from a 30 seconds excerpt of each song. The songs were manually annotated with six classes of emotions by three human experts.

---

<sup>19</sup> Originally from the UC Berkeley Enron Email Analysis at [http://bailando.sims.berkeley.edu/enron\\_email.html](http://bailando.sims.berkeley.edu/enron_email.html)



---

### 2.9.1.3 Biology

The learning task in yeast (Elisseeff and Weston 2001), one of the first non-textual and hence most popular multilabel classification problems, is to associate genes with a subset of 14 functional classes from the Comprehensive Yeast Genome Database of the Munich Information Center for Protein Sequences<sup>20</sup>. Each of the 2417 genes is represented with 103 features.

The last dataset in the list, *genbase*, contains a protein classification task (Diplaris et al. 2005). It consists of 663 protein chains represented by a vocabulary of 1186 short amino acid chains and sequence alignments associated with 27 distinct functional families.

### 2.9.2 Sources and Repositories

The Reuters *rcv1* dataset was retrieved from the online appendix of the article of Lewis et al. (2004), the older *r21578* from Lewis (2004). The *EUR-Lex* text collection is available at <http://www.ke.tu-darmstadt.de/resources/eurllex/> (Loza Mencía and Fürnkranz 2010). The remaining benchmark datasets were mainly retrieved from the very complete repository (Tsoumakas 2012) of the Mulan Java Library for Multi-Label learning (Tsoumakas et al. 2011b). Other resources include the repository (Chang and Lin 2012) of the LibSVM library for support vector machines (Chang and Lin 2001) and the collection accompanying the multilabel extension of WEKA (Read 2012). Other available datasets which were not employed in this work include the popular *yahoo* text classification benchmark (Ueda and Saito 2002) and the 2802 abstracts mapped to 1093 topics from high energy physics (Montejo Ráez et al. 2004).

## 2.10 Statistical Comparison of Classifiers

In Section 2.7, we saw how to evaluate the effectivity of a classifier and Section 2.7.2 showed how to maximally exploit the sparsely available data by using cross validation. To draw conclusions between classifiers simply by comparing the averaged metrics or even comparing results between different datasets is not reliable at all. This section describes statistical tests used in experimentation in order to validate the found differences.

For convenience, we will assume two classifiers *A* and *B* applied on a sequence of samples  $s_j$ ,  $j = 1 \dots N$ , in our case test instances or test sets, resulting in a sequence of (paired, and independent and identically distributed) observations  $\delta_j^A, \delta_j^B$ ,  $j = 1 \dots N$ , i.e. in the context of machine learning experimentation the metric scores on the test cases. From the (averaged) observations one can easily determine a dominance relation between both classifiers characterizing whether *A* outperformed *B* or the opposite (intuitively, no further tests are necessary if *A* and *B* tied). Roughly speaking, statistical tests allow us to determine a probability  $p$  that the difference was simply produced by chance. Or conversely, that the difference was not produced by chance with a probability of  $1 - p$ . If this

---

<sup>20</sup> <http://mips.gsf.de/genre/proj/yeast/>

---

significance value  $p$  is below an upper bound  $\alpha$ , the results are validated and the differences are determined to be reliable and *statistically significant* under a *significance level* of  $\alpha$ . Usually accepted values for  $\alpha$  are 1%, 5% or 10%.<sup>21</sup>

We describe three different test approaches which are all non-parametric as recommended by Demšar (2006), i.e. we do not assume any particular distribution on the observations. Other popular approaches such as ANOVA assume the normal distribution, which may result in a too inaccurate simplification.

The very simple sign test compares two classifiers and only counts the signs of the differences between the observations. It is recommended if observations are not comparable between samples and for a high number of samples. However, if the number of samples is low we may switch to the more powerful<sup>22</sup> Wilcoxon signed-ranks test, which additionally takes the magnitude of the differences into account. The differences should be comparable and commensurate between samples, and hence the test is especially recommended for cross validation experiments. For comparisons between multiple classifiers on multiple datasets, we describe the Friedman test and its post-hoc tests. The requirement here is a strict independence of the samples, hence no observations from same datasets such as in cross validation are allowed.

It is easy to get carried away from the result of a statistical test and hence draw the wrong conclusions. A statistical test is closely tied to the actual experimental setting and one has to be very careful if the results shall be transferred to different settings. In fact, the presented statistical tests do not allow any statement about even the repetition of tests, i.e. tests performed with samples drawn from the same population. Even if experiments were performed on multiple distinct datasets, and much more if only one dataset was used (sign test), a generalized transferability of the results is difficult to substantiate.

Another pitfall appears if no significance was detected. Roughly speaking, it is only possible to detect the presence but not the absence of statistical significance. E.g. every test fails for a specific small number of observations, and independently of the magnitude of the differences, but no conclusions in any direction can actually be drawn (only) from these failed tests. Moreover, statistical significance does generally not imply practical significance, and vice versa. This discussion is continued e.g. by Demšar (2008).

### 2.10.1 Sign Test

A sign test is a very simple non-parametric test in order to compare paired observations. Hence, it can be used e.g. to compare two classifiers that were applied to a sequence of test sets when the computed absolute metric scores are not comparable between different test sets. Consequently, the sign test only counts the number of samples  $N^A$  for

---

<sup>21</sup> Formally speaking, we assume the null hypothesis that the observation sequences have the same distribution (two-sided test).  $p$  denotes the probability that the observations were obtained under this hypothesis. If  $p$  is below a small significance quantile  $\alpha$ , the null hypothesis becomes implausible and is rejected. We can then assume that the observations are differently distributed.

<sup>22</sup> The power of a statistical test is the probability of detecting significance when there is actually significance. Less powerful tests commit more false negatives.



which a first algorithm was better (wins) and the number of times  $N^B$  where a second algorithm was better (losses). If both algorithms perform equally,  $N^A$  and  $N^B$  should be distributed according to the binomial distribution  $B(1/2; N)$ . If there is a significant deviation from this distribution, we can declare the performance of both algorithms statistically significantly different according to the sign test.

This simplest version of the sign test ignores the situation where both algorithms perform equally, i.e. there are tied observations. Three basic strategies exist for dealing with ties: 1) one half of the ties  $N^0/2$  is added to both  $N^A$  and  $N^B$ , respectively, 2) ties are omitted and a distribution according to  $B(1/2; N - N^0)$  is assumed for  $N^A$  or  $N^B$ , respectively, or 3) the ties are randomly distributed between both algorithms (cf. [Bian et al. 2011](#), [Coakley and Heise 1996](#), [Putter 1955](#)).

Following the second strategy, [Putter \(1955\)](#) proposed an *asymptotic uniformly most powerful non-randomized test* (ANU sign test) which is particularly suited for problems with a large proportion of ties. Assuming a large number of samples  $N > 25$  the test statistic

$$z = \frac{N^A - N^B}{\sqrt{N^A + N^B}} \quad (2.57)$$

is asymptotically distributed like the standard normal distribution  $N(0; 1)$ . Hence, both algorithms are statistically significantly different with a  $p$ -value of

$$p = 1 - \Phi(z) \quad (2.58)$$

and  $\Phi$  as the cumulative standard normal distribution function.

A sign test is simple, it does not make any assumptions about the distribution of the observations, and it is quite conservative (significance with the sign test implies significance with more sensitive tests but not vice versa). The ANU test showed a high *asymptotic relative efficiency* and was therefore recommended in a comparative study of several sign tests that allow for the possibility of large numbers of ties ([Coakley and Heise 1996](#)). [Fong et al. \(2003\)](#) proposes a modified, more powerful version of the test and a recent work by [Bian et al. \(2011\)](#) uses a trinomial distribution that models ties directly and hence allows for even more powerful tests.

However, the ANU test is not appropriate if the number of samples is low. E.g., in the case of 10 fold cross validation Eq. 2.57 maximally results in a  $p$ -value of 5.7% at  $N^A = 10$  and exact tables begin at  $N^A = 8$  ( $p = 5.5\%$ ). In these situations, we use instead the following more suitable and powerful test.

### 2.10.2 Wilcoxon Signed-Ranks Test

The Wilcoxon signed-ranks test ([Wilcoxon 1950](#)) has emerged as a non-parametric alternative to the still popular paired Students t-test, which shows several important weaknesses in the context of machine learning evaluation ([Demšar 2006](#)). In comparison to

the simpler sign test, it takes the magnitude of the differences into account, leading to an incremented expressiveness, i.e. power. On the other hand this requires comparability of the differences  $\Delta_j := \delta_j^B - \delta_j^A$ . Hence, strictly speaking, samples should be drawn from the same population, which is the case for cross validation.

For the Wilcoxon test we have to sort the paired observations according to their ascending absolute differences, i.e.  $|\Delta_{\pi_1}| \prec \dots \prec |\Delta_{\pi_N}|$ . Then we count the ranks of the positive and negative differences

$$W^+ := \sum_{\Delta_{\pi_r} > 0} r \quad W^- := \sum_{\Delta_{\pi_r} < 0} r \quad W := \min(W^+, W^-) \quad (2.59)$$

and obtain the test statistic  $W$ . For small  $N < 25$ , the  $p$ -value can be looked up in tables,  $W$  and  $\alpha$  pairs are e.g. (14, 10%), (10, 5%), (5, 1%) for  $N = 10$ . Otherwise,

$$z = \frac{W - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \quad (2.60)$$

provides a viable approximation based on Eq. 2.58.

We assign average ranks for equal  $|\Delta_{\pi_j}| = |\Delta_{\pi_{j+1}}| = \dots$  and evenly assume  $|\Delta_j| > 0$  or  $< 0$  for zero differences, ignoring one observation and reducing  $N$  by 1 if the number of  $|\Delta_j| = 0$  is odd (Pratt 1959). This approach was found by Conover (1973) to be the asymptotically most efficient strategy if  $\Delta$  is uniformly distributed.

### 2.10.3 Friedman and Post-Hoc Tests

We use the methodology described by Demšar (2006) for the comparison of multiple classifiers on multiple datasets. First, we perform a Friedman test (Friedman 1937, 1940) to determine whether the classifiers all perform similarly. Let  $k$  be the number of tested classifiers, let  $r_j^i$  be the ranks of the observations  $s_j^i$  on the  $j$ -th test case,  $s_j^{\pi_1} \succ \dots \succ s_j^{\pi_k}$ ,  $r_j^i = x \leftrightarrow \pi_x = i$  (cf. Eq. 2.21, we assign average ranks in case of ties) and let  $r^i = \text{avg}_j r_j^i$  be the average rank over all  $N$  test cases for classifier  $h_i$ . The null hypothesis now states that all classifiers perform equivalently and so their average ranks should be equal. Under this null hypothesis, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)} \sum_i \left( r^i - \frac{k+1}{2} \right)^2 = \frac{12N}{k(k+1)} \left( \sum_i (r^i)^2 - \frac{k(k+1)^2}{4} \right)$$

has a chi-squared distribution with  $k - 1$  degrees of freedom, when  $N$  and  $k$  are sufficiently large.

If the null hypothesis is rejected, we can determine which classifiers are significantly better than others with a post-hoc test. Demšar (2006) proposes to use the Nemenyi test

---

(Nemenyi 1963). The test entails that the performance of two classifiers is significantly different if the difference between their average ranks is at least the critical difference:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

where  $q_\alpha$  are critical values based on the Studentized range statistic divided by  $\sqrt{2}$ .

Demšar also describes the more powerful Bonferri-Dunn, Holm, Hochberg and Hommel tests which are based on pairwise comparisons between classifiers and used in comparisons against a control classifier. They all rely on the test statistic  $z = (r^i - r^j) / \sqrt{\frac{k(k+1)}{6N}}$  and a corresponding division of the resulting  $p$ -value. However, García and Herrera (2008) legitimate these tests also for performing all pairwise comparisons and in addition recommend the more powerful and complex Shaffer and Bergmann-Hommel tests. Details on the computation can be found in the cited publications.



---

## 3 Decompositive Approaches to Multilabel Classification

Decompositive approaches transform an original problem into several subproblems of a different type which can hopefully be solved more easily, more efficiently, or more effectively. In multilabel learning, the main purpose is to enable the use of existing state-of-the-art base learners. However, the choice of the decompositive approach certainly has an impact on efficiency, scalability and effectivity.

The predominant approach in multilabel classification is *binary relevance* learning (cf. Section 3.1). It tackles a multilabel problem by learning one classifier for each class, using all objects of this class as positive examples and all other objects as negative examples. Pairwise decomposition in contrast learns one classifier for each pair of classes. These pairwise classifiers are only trained in order to distinguish the corresponding two classes (cf. Section 3.4).

Tsoumakas and Katakis (2007) and Tsoumakas et al. (2010) classify both methods as problem transformation approaches. This also includes approaches which transform the problem into one single "sub"-problem, which is however different from the original one. We additionally make the following two distinctions: A transformational (or decompositive) approach is a *binarization* technique if the resulting subproblem(s) are (is) binary (cf. Section 2.5.1).

A further interesting property of decompositive approaches is the extent of the resulting subproblems, specifically whether they contain all points in the original data or only a subset. In binary relevance decomposition e.g. the union of the positive and negative examples of the subproblems results in the whole training set. This type of approaches are dedicated to separating a subspace from the whole instance space  $\mathcal{X}$ , whereby the subspace is represented by a set of positive examples and all the remaining known examples are assumed to be outside of this particular subspace, i.e. in the inverted subspace. Typically the presence of a particular property or characteristic is associated with the positive examples. We may hence consider the decomposition into *comprehensive subproblems* as a formalization of *concept learning*. This connection is further discussed in Section 3.1.

The most well-known counter-example for non-comprehensive subproblem generating approaches is pairwise decomposition, where a subproblem contains examples of two classes, and only these. The second class is not the inversion of the first class, as in binary relevance, but represents itself an explicitly given subspace. Hence, the subproblems only cover a subspace of  $\mathcal{X}$ .

Error correcting output codes (ECOC, Section 3.3) produce comprehensive subproblems, while the more general ternary ECOCs do not (cf. Section 3.5.8). Section 3.5.3 discusses tri-class learners and variations which are also non-comprehensive. An overview

**Table 3.1:** Overview of transformation approaches and properties. The total number of labels is denoted with  $n$  and the number of the examples in the training set with  $m$ .

transformation approach	number of subproblems	type of subproblems	extent of subproblems	Section
binary relevance decomposition	linear ( $n$ )	binary	comprehensive ( $m$ )	3.1
label powerset transformation	one	multiclass	comprehensive ( $m$ )	3.2
error correcting output codes	arbitrary ( $\geq n$ )	binary	comprehensive ( $m$ )	3.3
ternary ECOCs	arbitrary ( $\geq n$ )	binary	subset ( $< m$ )	3.5.8
pairwise decomposition	quadratic ( $\frac{n(n-1)}{2}$ )	binary	subset ( $< m$ )	3.4

of the main properties of the different approaches presented in this chapter is given in Table 3.1. [Boutell et al. \(2004\)](#) and [Tsoumakas and Katakis \(2007\)](#) list some additional transformation methods based on training example omission and repetition which are commonly not employed in practice (anymore). Section 9.3 and Figure 9.2(b) also shortly cover one of these methods (MC) in the context of feature selection.

### 3.1 Binary Relevance Decomposition

In the *binary relevance* (BR) decomposition, also known as *one-against-all* (OAA) or *one-against-the-rest/others* particularly for multiclass classification, a multilabel training set with  $n$  possible classes is decomposed into  $n$  binary training sets of the same size  $m = |\mathcal{T}_{rain}|$  that are then used to train  $n$  binary classifiers.

So for each input example  $(\mathbf{x}_j, \mathbf{y}_j)$  in the original training set  $\mathcal{T}_{rain}$ ,  $n$  different examples of the form  $(\mathbf{x}_j, y_{j,i})$  with  $i = 1 \dots n$  are generated resulting in the binary sets

$$\mathcal{T}_{rain}^i = \langle (\mathbf{x}_1, y_{1,i}), \dots, (\mathbf{x}_m, y_{m,i}) \rangle, i = 1 \dots n \quad (3.1)$$

with the new binary label output space  $y_{j,i} \in \mathcal{Y}_{bin}$ . Note again that all of these  $n$  decomposed training sets are of the same size as the original training set. A brief visual description of this technique is available in Figure 3.1.

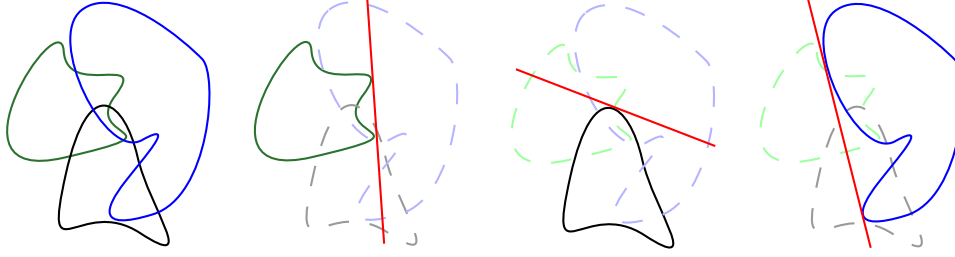
Hence,  $n$  different  $h_i = h_{\mathcal{T}_{rain}^i}$  binary base classifiers are trained in order to determine the relevance of  $\lambda_i$ , i.e. to recognize if an instance is included in their respective class  $\lambda_i$ . In consequence, the combined prediction of the binary relevance classifier for a test instance  $\mathbf{x}$  would be the output vector

$$\hat{\mathbf{y}} := (h_1(\mathbf{x}), \dots, h_n(\mathbf{x})) \quad (3.2)$$

or alternatively, in set notation,

$$\hat{P} := \{\lambda_i \mid h_i(\mathbf{x}) = 1\} \quad (3.3)$$

As already pointed out in Section 2.5.4, many base classifiers produce some type of relevance scores, which can be used to compare and rank classes. No assumption has to



**Figure 3.1:** Subproblems in *binary relevance classification* for multilabel classification: original three-class problem (green, blue and black classes, shown as overlapping clouds in left picture) is divided into green vs. rest (second picture), black vs. rest (third) and blue vs. rest two-class subproblems. Separating hyperplanes, denoted by red lines, have to respect all examples (inside the clouds). Clouds of negative examples have dotted lines.

be made at this point about the value range of the score predictions, though mostly the range is either  $[0; 1]$  or  $[-1; 1]$ . We will denote these predictions with  $h' : \mathcal{X} \rightarrow \mathbb{R}$ . The binary prediction  $h$  is obtained by means of thresholding at the middle  $\theta$  of the range<sup>23</sup>

$$t_\theta(s) := I[s > \theta] \quad , \quad s, \theta \in \mathbb{R} \quad (3.4)$$

and hence, we can define the following relationship for scoring classifiers

$$h = t_\theta \circ h' \quad (3.5)$$

If the context is clear, we may simply write  $h$  instead of  $h'$ .

Consequently, we define a prediction score vector as

$$\hat{\mathbf{v}} := (h'_1(\mathbf{x}), \dots, h'_n(\mathbf{x})) \in \mathbb{R}^n \quad (3.6)$$

and the corresponding sorted label ranking as<sup>24</sup>

$$\mathbf{r} = \langle \pi_1, \dots, \pi_n \rangle \in \Pi_{n,n}^n \quad \forall \pi_i, \pi_j, i < j. \quad h'_{\pi_i}(\mathbf{x}) > h'_{\pi_j}(\mathbf{x}) \quad (3.7)$$

Note however, that  $\theta$  must not be 0.5, although it is the most natural value since it was the objective to decide at that point. But, in fact, many approaches exist in order to select deviating thresholds and even to choose a different  $\theta_i$  for each classifier  $h'_i$ . A review can be found in Section 3.5.4.

There exists a strong connection between binary relevance decomposition and concept learning. As already detailed in Section 2.5.1, concept learning is dedicated to learning

<sup>23</sup> The use of  $>$  or  $\geq$  is arbitrary in practice, however, since we predict whether a label is relevant, it is more consistent to interpret a tie as neither relevant nor irrelevant than as relevant *and* irrelevant, and this is achieved with the present definition

<sup>24</sup> Without loss of generality, we assume that no ties happen in order keep the more elegant definition of  $\mathbf{r}$  as a total order. In practice, we may break ties randomly or according to the prior probability of the labels.

---

the presence or absence of a specific concept among instances. When several target concepts are possible or given for the same set of instances, we formally have a multilabel problem. In fact, there was an early understanding of the multilabel setting in the research field of concept learning. The first multilabel classification system known to the author, namely the *Construe topic identification system* used by Reuters (Hayes and Weinstein 1991), followed the paradigm of concept definitions and used a separate rule base for recognizing each label. Binary relevance could be seen as a solution to multilabel learning using concept learning since it decomposes a multilabel task into several subproblems which can be semantically considered concept learning problems. However, this is not the only valid possible interpretation. E.g., remind the example of the recognition of the color of an object in Section 2.5.1. Let the first (binary) class  $y_1$  in  $\mathcal{Y}$  determine whether an object is red or blue, and let the second class  $y_2$  represent either a circular or a rectangular object, etc. The resulting subproblems using binary relevance decomposition cannot be interpreted as concept learning problems anymore since there are no clear positive examples. Hence, binary relevance should be seen as a formally defined method in order to decompose multilabel problems in binary problems, regardless of the base learner used and the semantics of the labels and of the resulting binary problems. Decomposition according to concept learning is an instantiation of it which may be used if convenient.

The system of Joachims (1998) is among the first works known to the author which explicitly employed the generalized binary relevance decomposition approach with support vector machines for the binary subproblems. This method was later called *the binary approach* by Elisseeff and Weston (2001) and (curiously) *cross-training* by Boutell et al. (2004) until Brinker et al. (2006) coined the term used in this work.

### 3.1.1 Computational Complexity

The analysis of the computational complexity of BR uses the following variables: the total number of classes  $n$  and the training size  $m = |\mathcal{T}_{rain}|$ . In addition, we use the following two algorithm dependent variables for our extended analysis: Let  $p$  denote the training complexity grade with respect to the number of training examples, i.e. let the training computational complexity be  $\mathcal{O}(m^p)$ . Analogously, let the testing complexity for a single test example be  $\mathcal{O}(m^q)$ . In addition, for simplicity and because both values often correspond, we will also use  $q$  for declaring the model size  $\mathcal{O}(m^q)$ .

Some examples clarify  $p$  and  $q$ . Naive Bayes is linear in the number of training examples, i.e.  $p = 1$ , and  $q = 0$  as for all linear classifiers that explicitly store the separating hyperplane. Support vector machines usually behave super-linearly, i.e.  $p > 1$  (cf. Section 4.1.6). The testing efficiency depends on the number of support vectors, which is  $\mathcal{O}(m)$ , hence  $0 \leq q \leq 1$ , except for linear SVMs. k-Nearest Neighbor is a clear representative of the case of  $p = 0$  and  $q = 1$  (cf. Witten and Frank 2005, Sec. 3.8).

The variables  $p$  and  $q$  are not necessarily constant for one particular learning algorithm. They often strongly depend on the specific training data: difficulty of separation, balance of positive and negative examples, sparsity of features, etc. Even in the same domain



$p$  and  $q$  may not coincide. Especially for BR the complexity may change from (equal-sized) subproblem to subproblem. However, for the sake of simplicity of the analysis and comparisons, we will assume constant  $p$  and  $q$ , respectively, and e.g. interpret them as average complexity over all possible subsets of the same size on the particular training set (see also Fürnkranz 2002, Footnote 8). In addition, we will always provide an analysis that does not rely on these simplifying assumptions.

### Training

BR replicates each training example  $n$  times for using it for training each of the  $n$  classifiers. Hence, it follows from Eq. 3.1 that BR uses  $n \cdot m$  training examples. Since each classifier is trained with the full training set and using the extended notation, the complexity for training on  $\mathcal{T}_{rain}$  is  $\mathcal{O}(nm^p)$ .

### Predicting

Binary relevance has to store one classifier for each class, i.e.  $n$  models. The memory consumption of each model depends on the base learner, hence the complexity increases to  $n \cdot \mathcal{O}(m^q) = \mathcal{O}(nm^q)$ . Usually we will have to add at least  $\mathcal{O}(m)$  space costs during training for the storage of the training data itself.

The prediction costs change analogously. BR has to evaluate  $n$  classifiers for each test example, hence testing an example costs  $\mathcal{O}(nm^q)$  operations.

## 3.2 Label Powerset Transformation

In the *label powerset* approach (LP), a meta multiclass problem is constructed, where each appearing label combination  $P_j$  is interpreted as one separate class (cf. Boutell et al. 2004, Tsoumakas and Katakis 2007). The meta problem is then solved with a normal multiclass algorithm or with the decomposition methods presented in this chapter. Hence, the core of LP is formally not a decompositive nor a binarization approach, but only a special form of problem transformation. However, LP is often combined with a binarization step, in particular one-against-all.

Formally, the resulting single-label multiclass problem has the following properties:

$$\mathcal{L}_{LP} := \{P_j \mid 1 \leq j \leq m\} = \{\lambda'_1, \dots, \lambda'_{n'}\} \in 2^{\mathcal{L}}, \quad n' = |\mathcal{L}_{LP}| \quad (3.8)$$

$$\mathbf{y}' = (y'_1, \dots, y'_{n'}) \in \mathcal{Y}_{mc}^{n'}, \quad y'_i = \begin{cases} 1, & \text{if } P = \lambda'_i \\ 0, & \text{otherwise} \end{cases} \quad (3.9)$$

$$\mathcal{T}_{rain}^{LP} := \langle (\mathbf{x}_1, \mathbf{y}'_1), \dots, (\mathbf{x}_m, \mathbf{y}'_m) \rangle \quad (3.10)$$

The predictions  $\hat{\mathbf{y}}' = h_{\mathcal{T}_{rain}^{LP}}$  can be directly processed: the classifier predicts the labelset  $P_j = \lambda'_i$  with  $\hat{\lambda}'_i = 1$ . This also demonstrates that only labelsets that were present in the training set can be predicted for the test set. The higher the number of labels and the label density, the higher the probability that each example has a distinct labelset associated (cf. Table 2.2), and hence the lower the probability that the LP approach is

able to predict the correct labelset on a test set. However, given that the new labelsets in the test set are similar to known labelsets in the training set, this scenario does usually not prevent LP of being able to obtain good (yet not perfect) results and even outperform other approaches (except of course for subset accuracy, cf. Section 2.7.3).

No ranking can be directly computed under normal circumstances. However, it is possible for probabilistic base classifiers (cf. Section 2.6).

### 3.2.1 Computational Complexity

In the worst case, the resulting multiclass problem has an increased amount of classes of  $n' = |\{P_i \mid 1 \leq i \leq m\}| = \min(m, 2^n)$ , where  $m$  is the number of training examples. This number tends to be much lower than the given bound for many real world datasets (evident counterexamples are *hifind*, *delicious*, *EUROVOC* in Table 2.2, but even then the number of powerset classes may be prohibitively high.

The particular complexities for training, testing and memory then depend on the employed multiclass solver. For one-against-all and pairwise decomposition (with  $d = 1$ ) e.g. the analyses given in Sections 3.1.1 and 3.4.6 apply.

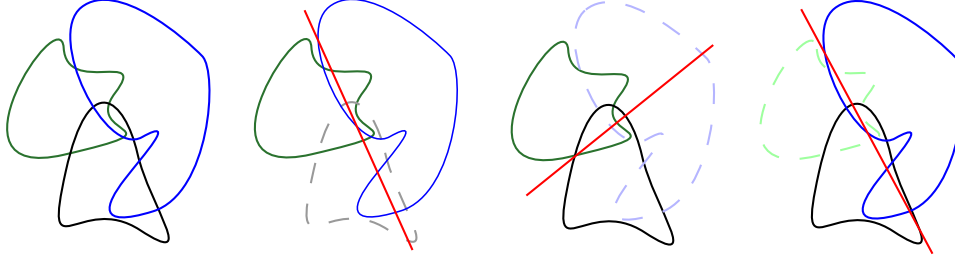
## 3.3 Error Correcting Output Codes

Error-correcting output codes (ECOC) is a generalized framework for binary decomposition which potentially allows any type of arbitrary and also randomized decomposition (Dietterich and Bakiri 1995). A decomposition scheme is represented as a matrix with the classes at the rows and the base classifiers at the columns. A 1 in a cell means that positive examples of the corresponding class are also used as positive examples for the corresponding classifier, while a  $-1$  means that the examples are used as negatives. As an example, the  $n \times n$  matrix of the one-against-all classifiers for the problem depicted in Figure 3.1 would be

$$\begin{matrix} & h_1 & h_2 & h_3 \\ \lambda_1 & \begin{pmatrix} 1 & -1 & -1 \end{pmatrix} \\ \lambda_2 & \begin{pmatrix} -1 & 1 & -1 \end{pmatrix} \\ \lambda_3 & \begin{pmatrix} -1 & 1 & 1 \end{pmatrix} \end{matrix} \quad (3.11)$$

However, ECOC starts from the premise of dynamically filled matrix, e.g. randomly. During prediction, a vector is filled with 1 or  $-1$  according to the predictions of the binary classifiers. The row with the smallest distance (e.g. Hamming, cf. Eq. 2.43) is determined and the corresponding label is chosen.

Allwein et al. (2000) extend the framework to ternary matrices allowing also zeros. If a cell contains a zero, the examples from the associated examples are just ignored. This makes it possible to represent the pairwise decomposition introduced in the next section, shown in Eq. 3.34.



**Figure 3.2:** Subproblems in *pairwise multilabel classification*: original three-class problem is divided into green vs. blue (second picture, black examples are ignored), green vs. black (blue is ignored) and blue vs. black two-class subproblems. Separating hyperplanes have to respect only examples from two classes in contrast to BR in Figure 3.1. Dotted lines denote the ignored class.

### 3.4 Pairwise Multilabel Decomposition

*Pairwise learning* came up as an alternative approach to one-against-all binarization for multiclass classification. While one-against-all is essentially concerned with the individual detection of particular classes, which is effectively achieved by considering all other known examples as not belonging to the respective class, the main idea of pairwise learning is to learn to discriminate between pairs of classes. Hence, other used denominations are *one-against-one*, *all-against-all*, or *all-pairs* in addition to *pairwise classification/decomposition/binarization*, *round robin classification* (Fürnkranz 2002) and *learning by pairwise comparison* (LPC).

We divide the explanation of pairwise learning into two parts: the next section introduces the decomposition of the original problem by first reviewing pairwise decomposition for multiclass classification, Section 3.4.3 describes the combination of the base classifiers to form an overall prediction. A general analysis of the computational complexity is given in Section 3.4.6.

#### 3.4.1 Decomposition

As already shortly sketched, one classifier is trained for each pair of classes in pairwise learning, i.e., a problem with  $n$  different classes is decomposed into  $\frac{n(n-1)}{2}$  smaller binary subproblems. For each pair of classes  $(\lambda_u, \lambda_v)$ , only examples belonging to either  $\lambda_u$  or  $\lambda_v$  are used to train the corresponding base classifier (BC)  $h_{u,v}$ . All other examples are ignored.

In the multilabel case, and assuming  $u < v$ , the process is equivalent: An example is added to the training set for classifier  $h_{u,v}$  if  $\lambda_u$  is a relevant class and  $\lambda_v$  is an irrelevant class or vice versa, i.e., if  $(\lambda_u, \lambda_v) \in P \times N$  or vice versa  $(\lambda_u, \lambda_v) \in N \times P$  with  $N = \mathcal{L} \setminus P$  as negative labelset (cf. Figure 3.2). Thus training examples of class  $\lambda_u$  will receive a training signal of 1, whereas training examples of class  $\lambda_v$  will be classified with 0.

More formally, the binary subproblems are generated as follows:

$$\mathcal{T}_{rain}^{u,v} := \langle (\mathbf{x}, y_u) \in \mathcal{T}_{rain} \times \mathcal{Y}_{bin} \mid y_u + y_v = 1 \rangle \quad 1 \leq u < v \leq n \quad (3.12)$$

Note that  $y_u + y_v = 1$  is simply a compact formulation of the equivalent expression  $(y_u = 1) \text{ XOR } (y_v = 1) = (y_u = 1) \wedge (y_v = 0) \vee (y_u = 0) \wedge (y_v = 1)$  or  $(\lambda_u, \lambda_v) \in P \times N \vee (\lambda_u, \lambda_v) \in N \times P$  exploiting our convention in Section 2.4 and Eq. 2.12. The statement shows the decomposition from the perspective of a classifier  $h_{u,v} = h_{\mathcal{T}_{rain}^{u,v}}$  and is reflected in Figure 3.2. Note again that  $\mathcal{T}_{rain}^{u,v}$  only contains examples for which the label assignments  $y_u$  and  $y_v$  are distinct.

The following equation shows the decomposition from the perspective of a training example  $(\mathbf{x}, P)$ . In particular, it shows for which subproblems the input vector  $\mathbf{x}$  is considered a positive or negative example.

$$\begin{aligned} (\mathbf{x}, 1) \in \mathcal{T}_{rain}^{u,v} & \iff \lambda_u \in P, \lambda_v \in N, u < v \\ (\mathbf{x}, 0) \in \mathcal{T}_{rain}^{u,v} & \iff \lambda_v \in P, \lambda_u \in N, u < v \end{aligned} \quad (3.13)$$

This corresponds to the illustration in Figure 3.3, which also shows pairwise learning from the point of view of *preference learning* and is introduced in the following.

### 3.4.2 Pairwise Preference Learning

*Preference learning* (cf. Section 2.5.4), which aims at modeling and learning preferences between a set of alternative objects, provides a suitable basis for the decomposition approach described in the previous section and hence justifies its use. More specifically, we stick to the framework of *pairwise preference learning* (Fürnkranz and Hüllermeier 2003, Hüllermeier et al. 2008), which focuses particularly on decomposing problems into fine-grained preference relations between pairs of objects. To this effect, we break down a multilabel problem into preference statements between pairs of classes and try to learn these.

A classifier  $h_{u,v}$  is trained in order to detect the preference  $\lambda_u \succ_{\mathbf{x}} \lambda_v$  or  $\lambda_v \succ_{\mathbf{x}} \lambda_u$  for a particular instance  $\mathbf{x}$ . Obviously, we can only state a preference relation between  $\lambda_u$  and  $\lambda_v$  for an instance  $\mathbf{x}$  from multilabel data if both labels are not contained in the same set  $P_{\mathbf{x}}$  or  $N_{\mathbf{x}}$ , since no distinction can be inferred between labels inside one of these two labelsets. The only available information is that labels in  $P_{\mathbf{x}}$  are relevant and hence preferred over the labels in  $N_{\mathbf{x}}$ , which are irrelevant for the particular instance.

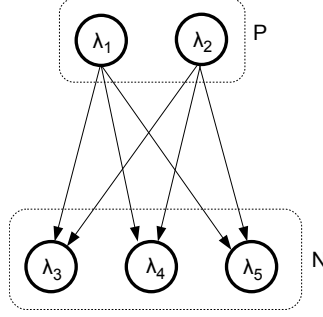
Based on Section 2.5.4 and Eq. 2.22, we see multilabel classification as a label ranking problem with bipartite rankings  $\mathbf{r} = \langle P, N \rangle$  and hence define the preference relation on  $\mathcal{L}$  formally as follows:

$$\lambda_u \succ_{\mathbf{x}} \lambda_v \iff \lambda_u \in P_{\mathbf{x}} \wedge \lambda_v \in N_{\mathbf{x}} \quad (3.14)$$

Correspondingly, we can generalize the training of the base classifiers in Eq. 3.12 by

$$\mathcal{T}_{rain}^{u,v} := \langle (\mathbf{x}, y_u) \in \mathcal{T}_{rain} \times \mathcal{Y}_{bin} \mid y_u \succ_{\mathbf{x}} y_v \vee y_u \prec_{\mathbf{x}} y_v \rangle \quad (3.15)$$

Due to the strictness and totality of  $\succ_{\mathbf{x}}$ , instances with  $\lambda_u, \lambda_v \in P$  are ignored by  $h_{u,v}$ . This point is discussed in more detail in Section 3.5.3.



**Figure 3.3:** Pairwise multilabel training: training example  $\mathbf{x}$  belongs to  $P = \{\lambda_1, \lambda_2\}$ ,  $N = \{\lambda_3, \lambda_4, \lambda_5\}$  are the irrelevant classes. The arrows represent the learned preferences, i.e. the trained classifiers (Fig. based on Brinker et al. 2006).

Note that we may break with the convention of  $u < v$  and assume symmetric training sets

$$\mathcal{T}_{rain}^{v,u} = \langle (\mathbf{x}, 1 - y) \mid (\mathbf{x}, y) \in \mathcal{T}_{rain}^{u,v} \rangle \quad (3.16)$$

If a learning algorithm does not produce symmetric classifiers, i.e.  $h_{u,v}(\mathbf{x}) \neq 1 - h_{v,u}(\mathbf{x})$  for all  $\mathbf{x} \in \mathcal{X}$ , such as for many rule learners, it is possible to explicitly train both  $h_{u,v}$  and  $h_{v,u}$  in order to eliminate a possible bias in the base learner (Fürnkranz 2002). However, we will assume symmetric classifiers in this work.

### 3.4.3 Aggregation

During classification, the predictions of the base classifiers  $h_{u,v}$  are interpreted as *preference statements* that predict for a given example which of the two labels  $\lambda_u$  or  $\lambda_v$  is preferred. We will use the term *aggregation* to refer to the combination of the pairwise *base predictions* into an overall, global prediction. More specifically, we will only deal with approaches which generate a ranking over the labels. As an example, a simple strategy for combining the predictions into a ranking is known as *max-wins*, *plurality* or *majority voting* (Kuncheva 2004, Sec. 4.2). It interprets each binary preference as a vote for the preferred class, thus obtaining a sum of votes for each label. Aggregation is thus often also called *voting*.

Indeed, we can define aggregation as summation over votes: Without loss of generality, let  $a_{\mathbf{x}} : [0; 1] \rightarrow \mathbb{R}$  be a voting function, let  $h' : \mathcal{X} \rightarrow [0; 1]$  be a soft binary classifier with  $\theta = 0.5$  (cf. Section 3.1) that is symmetric so that  $h'_{u,v} = 1 - h'_{v,u}$ . Then the aggregation of the  $(n-1)n/2$  base predictions  $h'_{u,v}(\mathbf{x})$ ,  $1 \leq u, v \leq n$  into a voting vector  $\mathbf{v} \in \mathbb{R}^n$  according to  $a_{\mathbf{x}}$  is given by

$$v_u = \sum_{\substack{1 \leq v \leq n \\ v \neq u}} v_{u,v} = \sum_{\substack{1 \leq v \leq n \\ v \neq u}} a_{\mathbf{x}}(h'_{u,v}(\mathbf{x})) \quad (3.17)$$

A ranking  $\mathbf{r}$  over the labels can be obtained by ordering according to the descending sums of votes  $v_u$  so that  $r(u) < r(v) \Leftrightarrow v_u > v_v$ .<sup>25</sup> Usually, the voting function will be monotonically increasing, but even  $a_{\mathbf{x}}(0) \leq a_{\mathbf{x}}(1)$  cannot be generally guaranteed (e.g. for stacking based approaches). Note that  $a_{\mathbf{x}}$  is dependent on  $\mathbf{x}$  since some aggregation strategies may compute  $v_{u,v}$  in dependence of the other outcomes  $h'_{i,j}(\mathbf{x})$ ,  $i \neq v, j \neq v$ .

However, we will exemplarily focus on two simple strategies, which are the most widely used (cf. Galar et al. 2011, Hüllermeier and Vanderlooy 2010) and which compute  $a_{\mathbf{x}}$  independently of  $\mathbf{x}$ .

- **Simple Voting**

Also known as *0-1-voting*, *binary voting* or simply *voting*, uses the indicator function as  $a_{\mathbf{x}}$ , hence only *full votes* 0 or 1 are distributed to the classes. Obviously, simple voting does not need continuous predictions.

$$v_u = \sum_{\substack{1 \leq v \leq n \\ v \neq u}} I[h'_{u,v}(\mathbf{x}) > 0.5] = \sum_{\substack{1 \leq v \leq n \\ v \neq u}} h_{u,v}(\mathbf{x}) \quad (3.18)$$

- **Weighted Voting**

The output scores of the base classifiers are interpreted as a type of confidence or preference degree, and hence partial or *weighted votes* are dispensed. The voting function is the identity function.

$$v_u = \sum_{\substack{1 \leq v \leq n \\ v \neq u}} h'_{u,v}(\mathbf{x}) \quad (3.19)$$

Both approaches have in common that each base classifier has equal voting mass which can be distributed among two classes. Therefore, all labels can at most receive  $n - 1$  votes:

$$1 = a_{\mathbf{x}}(h'_{u,v}(\mathbf{x})) + a_{\mathbf{x}}(h'_{v,u}(\mathbf{x})) \quad v_u \leq n - 1 \quad 1 \leq u, v \leq n, u \neq v \quad (3.20)$$

Figure 3.4 shows a possible result of classifying the sample instance of Figure 3.3. Base classifier  $h_{1,5}$  predicts the first class (correctly). In consequence,  $\lambda_1$  receives one vote and class  $\lambda_5$  receives zero votes (denoted by  $h_{1,5} = 1$  in the first and  $h_{5,1} = 0$  in the last row). All 10 base classifiers (the values in the upper right corner can be deduced based on the assumption of symmetry) are evaluated, although only six are ‘competent’ since only those were trained with the original example.

A discussion on the competence aspect and on the appropriateness of voting strategies in general is found in Sections 3.5.2 and 3.5.6, respectively.

<sup>25</sup> In order to obtain a total ranking, if not otherwise stated, we break ties arbitrarily. Other approaches consist in ordering according to the apriori class probability, or in considering the base pairwise comparisons.

$h_{1,2} = 1$	$h_{2,1} = 0$	$h_{3,1} = 0$	$h_{4,1} = 0$	$h_{5,1} = 0$
$h_{1,3} = 1$	$h_{2,3} = 1$	$h_{3,2} = 0$	$h_{4,2} = 0$	$h_{5,2} = 0$
$h_{1,4} = 1$	$h_{2,4} = 1$	$h_{3,4} = 1$	$h_{4,3} = 0$	$h_{5,3} = 0$
$h_{1,5} = 1$	$h_{2,5} = 1$	$h_{3,5} = 1$	$h_{4,5} = 1$	$h_{5,4} = 0$
$v_1 = 4$	$v_2 = 3$	$v_3 = 2$	$v_4 = 1$	$v_5 = 0$

**Figure 3.4:** Pairwise voting: an example  $\mathbf{x}$  is classified by all 10 base classifiers  $h_{u,v}, u \neq v, \lambda_u, \lambda_v \in \mathcal{L}$ . Note the redundancy given by  $h_{u,v} = 1 - h_{v,u}$ . The last line counts the (positive) outcomes for each class.

### 3.4.4 Bipartitioning

As already mentioned, we obtain a ranking over the classes with the help of the vote vector  $\mathbf{v}$  in Eq. 3.17. Therefore, LPC in this form is also called *ranking by pairwise comparison* (Fürnkranz et al. 2008). We have learned in Section 2.5.4 that multilabel classification can be considered an instantiation of label ranking and is also often evaluated as such (cf. Section 2.7.4), and we will get to know the advantages of such a perspective in the discussion in Section 3.5.5. In fact, MLC is often reduced to ranking classes (i.e. the task of label ranking) in the literature (e.g. Crammer and Singer 2003, Elisseeff and Weston 2001).

Notably, a separation of predicted positive and negative labels can easily be produced by splitting the ranking at a certain position: all labels above the split are considered positive and all below are considered negative. This approach is called *thresholding* or more generally *bipartitioning*.

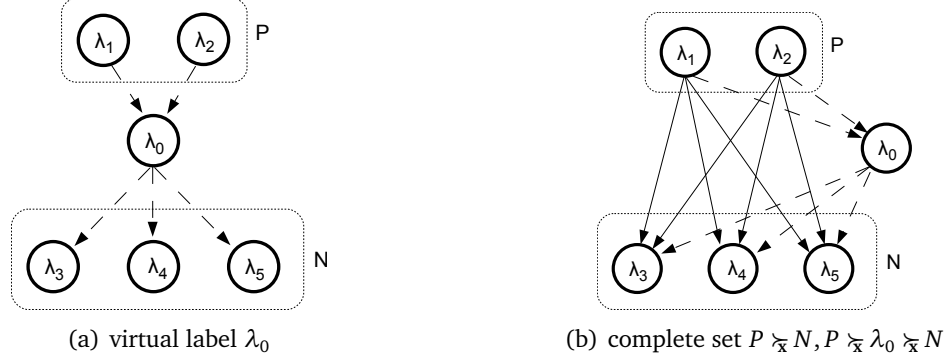
A general bipartitioning function was already introduced in Eq. 2.26. We specify the definition in order to use vote vectors

$$b : \mathbb{R}^n \rightarrow 2^{\mathcal{L}} \quad (3.21)$$

Other approaches rely on the produced rankings or even on the input vectors  $\mathbf{x}$ , and use learners to learn these functions or simply use static thresholds. Some of the methods are revised in Section 3.5.4. But for the following technique the definition given above is sufficient.

### 3.4.5 Calibration

To convert the resulting ranking of labels into a multilabel prediction, we use the *calibrated label ranking* approach (Brinker et al. 2006, Fürnkranz et al. 2008). This technique avoids the need for learning a threshold function  $t$  for separating relevant from irrelevant labels. The key idea is to introduce an artificial *calibration label*  $\lambda_0$ , which represents the split-point between relevant and irrelevant labels. Thus, it is assumed to be preferred over



**Figure 3.5:** Calibrated pairwise multilabel learning: virtual label  $\lambda_0$  is introduced which separates  $P$  and  $N$ , i.e.  $\lambda_1, \lambda_2 \succ_x \lambda_0 \succ_x \lambda_3, \lambda_4, \lambda_5$ . Additionally trained base classifiers  $h_{1,0}, h_{2,0}, h_{0,3}, h_{0,4}, h_{0,5}$  correspond to BR's  $h_1, \dots, h_5$ . The whole set of learned preferences from Figure 3.3 and (a) is depicted in (b) (Fig. based on Brinker et al. 2006).

$h_{0,1} = 0$	$h_{1,0} = 1$	$h_{2,0} = 1$	$h_{3,0} = 0$	$h_{4,0} = 0$	$h_{5,0} = 0$
$h_{0,2} = 0$	$h_{1,2} = 1$	$h_{2,1} = 0$	$h_{3,1} = 0$	$h_{4,1} = 0$	$h_{5,1} = 0$
$h_{0,3} = 1$	$h_{1,3} = 1$	$h_{2,3} = 1$	$h_{3,2} = 0$	$h_{4,2} = 0$	$h_{5,2} = 0$
$h_{0,4} = 1$	$h_{1,4} = 1$	$h_{2,4} = 1$	$h_{3,4} = 1$	$h_{4,3} = 0$	$h_{5,3} = 0$
$h_{0,5} = 1$	$h_{1,5} = 1$	$h_{2,5} = 1$	$h_{3,5} = 1$	$h_{4,5} = 1$	$h_{5,4} = 0$
$v_0 = 3$	$v_1 = 5$	$v_2 = 4$	$v_3 = 2$	$v_4 = 1$	$v_5 = 0$

**Figure 3.6:** Pairwise voting from Figure 3.4 extended with calibrating label  $\lambda_0$ : an example  $\mathbf{x}$  is classified by all 15 base classifiers. The last line counts the positive outcomes for each class. Only labels  $\lambda_1$  and  $\lambda_2$  would be ranked above the virtual label and hence predicted as relevant.

all irrelevant labels, but all relevant labels are preferred over  $\lambda_0$ . Essentially, we transform the bipartite ranking given by  $P$  and  $N$  into the tripartite ranking  $\mathbf{r}$  (cf. Section 2.5.4)

$$P \succ_{\mathbf{r}} \lambda_0 \succ_{\mathbf{r}} N \quad (3.22)$$

The introduction of the additional label during training is graphically depicted in Figure 3.5(a), the combination with the normal pairwise base classifiers is shown in Figure 3.5(b).

As it turns out, the resulting  $n$  additional binary classifiers  $\{h_{0,v} \mid v = 1 \dots n\}$  are identical to the classifiers that are trained by the binary relevance approach.<sup>26</sup> Thus, each classifier  $h_{0,v}$  is trained in a one-against-all fashion by using the whole dataset with

<sup>26</sup> Following Brinker et al. (2006) and Fürnkranz et al. (2008), if  $\lambda_u$  is relevant for  $\mathbf{x}$ , then  $\mathbf{x}$  is a positive training example for  $h_u$ , and a negative one for  $h_{0,u}$  since  $\lambda_u \succ_x \lambda_0$ . The opposite holds for  $\lambda_u$  being irrelevant. Due to the symmetry of the base learners,  $h_{0,u}$  equals  $1 - h_u$ .



$\{\mathbf{x} \mid \lambda_v \in P_{\mathbf{x}}\}$  as negative examples since  $\lambda_v \succ_{\mathbf{x}} \lambda_0$ , and  $\{\mathbf{x} \mid \lambda_v \notin P_{\mathbf{x}}\}$  as positive examples since  $\lambda_0 \succ_{\mathbf{x}} \lambda_v$ . This results in additional  $n$  training sets<sup>27</sup>

$$\mathcal{T}_{rain}^{0,v} := \langle (\mathbf{x}, 1 - y_v) \in \mathcal{T}_{rain} \times \mathcal{Y}_{bin} \rangle \quad 1 \leq v \leq n \quad (3.23)$$

At prediction time, we will thus get a predicted ranking  $\mathbf{r}$  over  $n + 1$  labels (the  $n$  original labels plus the calibration label) and the additional label will indicate the zero-point

$$\lambda_{\pi_1} \prec_{\mathbf{r}} \dots \prec_{\mathbf{r}} \lambda_{\pi_d} \prec_{\mathbf{r}} \lambda_0 \prec_{\mathbf{r}} \lambda_{\pi_{d+2}} \prec_{\mathbf{r}} \dots \prec_{\mathbf{r}} \lambda_{\pi_n} \quad (3.24)$$

with  $d = |\hat{P}|$ . With respect to the predicted voting vector, the projection to a pure multi-label output is similarly quite straightforward:

$$\hat{P} = t_{\lambda_0}(\mathbf{v}) := \{\lambda_v \mid v_v > v_0\} \quad (3.25)$$

As for solving ranking ties, we may throw a die for  $\lambda_v$  if  $v_v = v_0$  or use  $v_v \geq v_0$ , which favors recall.

Figure 3.6 extends the example from Figure 3.4 and shows a possible result of classifying with the calibrated label  $\lambda_0$ . It shows the ideal case, i.e. the base classifiers were trained according to Figure 3.5(b) and predict correctly. For instance, the relevant classes  $\lambda_1$  and  $\lambda_2$  may receive a vote, respectively, in direct comparison with the calibrated label. After evaluating all base classifiers, the number of votes for the calibrated label  $v_0$  is used as the split-point to discriminate relevant classes from irrelevant classes. In this example,  $\lambda_1$  and  $\lambda_2$  are returned as the set of relevant classes  $\hat{P}$ .

We denote the approach of using an additional virtual label in training the pairwise ensemble *calibrated label ranking* (CLR).

### 3.4.6 Computational Complexity

We rely for the analysis on the variables given in Section 3.1.1, i.e.  $m$  for the number of training examples,  $n$  for the number of possible labels,  $p$  and  $q$  for algorithm complexity. In addition, we introduce  $l = \sum_{\mathbf{x} \in \mathcal{T}_{rain}} |P_{\mathbf{x}}|$  as the total number of all relevant labels in the training set and  $d = l/m$  as the average labelset size in  $\mathcal{T}_{rain}$ .

The analysis on basis of the complexity factors  $p$  and  $q$  has to be taken with more care than in Section 3.1.1, since we idealistically assume class-balanced subproblems,<sup>28</sup>

<sup>27</sup> Following Eq. 3.1 and due to the symmetry of the base learners,  $h_{0,i} = 1 - h_i$  holds (Brinker et al. 2006, Fürnkranz et al. 2008).

<sup>28</sup> Note that subproblems of equal size are ideal for the analysis but not necessarily for the actual efficiency. It is not clear whether and under which circumstances a class-balanced scenario is more beneficial than unbalanced classes: unbalanced classes can lead to some (slow) large binary subproblems, but also to many small problems and unbalanced subproblems, which may be much easier and quicker to solve.

**Table 3.2:** Complexity comparison between BR and LPC with respect to used training examples and number of classifiers that need to be stored and that are queried while testing one example. Computational costs are given in terms of the number of classes  $n$ , training set size  $m$  and average labelset size  $d$ . The symbol  $\leq$  indicates (tight) bounds instead of exact calculation. For CLR the rows of BR and LPC have to be summed up.

	training examples	number of classifiers
BR	$nm$	$n$
LPC	$\leq nmd$	$n(n-1)/2 = \mathcal{O}(n^2)$
LPC/BR	$\leq d$	$n-1/2 = \mathcal{O}(n)$

which is almost never the case in practice. However, we are able to provide an intuitive and expectably valid estimation which allows to easily and briefly compare to other decomposition methods like BR. A more formal and more general analysis was done by Fürnkranz (2002) for simple multiclass pairwise decomposition. The analysis without taking into account the dependency on  $p$  and  $q$  is mainly based on the works by Brinker et al. (2006) and Fürnkranz et al. (2008) and was only slightly extended.

We will try to deduce exact values or close upper bounds, but we may provide estimations in Landau big  $\mathcal{O}$  notation where it seems opportune in order to emphasize the asymptotic behavior or because of a more compact presentation. While Table 3.2 shows the findings of Section 3.4.6.1 and 3.4.6.2, Table 3.3 summarizes the analysis under idealistic assumptions (cf. Section 3.4.6.3).

### 3.4.6.1 Training

In previous work, it has been shown that training a pairwise classifier requires  $\mathcal{O}(nm)$  training examples (Fürnkranz 2002). To see this, note that each of the  $m$  original training examples will only appear in the training sets of  $n-1$  models. Therefore the total number of training examples that have to be processed is  $(n-1)m$ .

For multilabel classification, pairwise decomposition will additionally compare a training example's  $|P|$  relevant labels to all  $|P| = n - |P|$  labels that are not relevant for this example.

Thus, each example occurs in

$$|P| \cdot |N| = |P|(n - |P|) = n|P| - |P|^2 \quad (3.26)$$

training sets. Let  $l = \sum_{\mathbf{x} \in \mathcal{T}_{rain}} |P_{\mathbf{x}}|$  be the total number of all relevant labels in the training set, let  $m = |\mathcal{T}_{rain}|$  be the training set size and  $d = l/m$  the average labelset size, then the total number of training examples in all training sets is

$$\sum_{\mathbf{x}} (n|P_{\mathbf{x}}| - |P_{\mathbf{x}}|^2) = n \cdot \sum_{\mathbf{x}} |P_{\mathbf{x}}| - \sum_{\mathbf{x}} |P_{\mathbf{x}}|^2 \leq nl - l = (n-1)l \quad (3.27)$$

with  $|P_x|^2 \geq |P_x|$ .<sup>29</sup>

Thus, the total number of training examples needed to train the  $n(n-1)/2$  base classifiers is  $\mathcal{O}(nl) = \mathcal{O}(nmd)$ . For the  $n(n+1)/2$  base classifiers of the calibrated version, we additionally need  $nm$  examples for training  $h_{0,v}$ ,  $1 \leq v \leq n$ , leading to the increased total number of  $(n-1)md + nm < nm(d+1) = \mathcal{O}(nmd)$ .

This result shows that the complexity of training the pairwise ensemble depends critically on the total number of relevant labels in the training examples. For conventional pairwise classification, i.e.  $l = m$ , the derived bound reduces to the linear  $\mathcal{O}(nm)$  bound that was shown by Fürnkranz (2002). Multilabel classification increases this bound by a factor of  $l/m$  to  $\mathcal{O}(nm \cdot l/m) = \mathcal{O}(nl)$ , i.e., the complexity of CLR is within a factor of  $l/n = d$  of the  $\mathcal{O}(nm)$  examples needed for training a BR classifier.

Thus, the crucial factor that determines the complexity of the approach is  $d$ , the average number of labels per example. We would like to point out that in many practical applications, this factor is determined by the procedure that is used for labeling the training examples, and is independent of  $n$ . A typical example is the number of keywords that are assigned to a text, which typically is a low number and does not depend on the number of available keywords. For example, in the *rcv1* text categorization dataset, the median value of keywords is three, and the percentage of documents with more than ten keywords is approximately one per thousand (cf. also Figure 4.5). Even for the highly dense *hifind*, only nearly 6% of the labels are set on average per instance (cf. Table 2.2).

Thus, for many practical applications, the complexity of pairwise decomposition is within a small constant factor  $d$  of BR's complexity. Of course, the worst-case complexity, which would occur when all possible labels are *a priori* equally likely for every example, is  $\mathcal{O}(n^2m)$ .

### 3.4.6.2 Predicting

The above results, that show that the effort is linear in the total number of labels in the training set, only apply to the training time. We still have to store a quadratic number of classifiers, and, in principle, all of them have to be queried at classification time. Even with methods reducing the number of base classifier evaluations (cf. Chapter 5), the quadratic storage complexity remains, which may become a practical burden for large number of classes.

Formally, the number of models that have to be stored in the original formulation of the pairwise ensemble is  $\frac{n(n-1)}{2}$  and this is also the amount of classifier evaluations. If CLR is used, we have to add  $n$  for the additional classifiers, which is certainly a smaller overhead than for the number of training examples used. For the training data itself and assuming an appropriate storage strategy using pointers, we require space for  $\mathcal{O}(m)$  training examples during training.

<sup>29</sup> Note that a slightly different estimation  $|P|(n - |P|) \leq |P| \cdot n$  is used by Brinker et al. (2006) and Fürnkranz et al. (2008), resulting in a slightly less tight bound  $\leq nl$ .

**Table 3.3:** Extended complexity comparison under idealistic assumptions of equal-sized classes with  $dm/n$  examples each and non-linear or non-constant dependency, respectively, on the number of training examples. Additional assumptions are  $1 \leq p \leq \hat{p}$  for non-linear training and  $0 \leq q \leq \hat{q} \leq 1$  for non-constant testing and memory. The training cost on the complete training set  $\mathcal{T}_{rain}$ , the prediction costs on *one* test example and the total memory space are given for BR, LPC and each of their base classifiers (BCs) in terms of number of classes  $n$ , training set size  $m$  and average labelset size  $d$ .

	training time	prediction time & memory requirements
BR's BCs	$\mathcal{O}(m^{\hat{p}})$	$\mathcal{O}(m^{\hat{q}})$
BR	$\mathcal{O}(nm^{\hat{p}})$	$\mathcal{O}(nm^{\hat{q}})$
LPC's BCs	$\mathcal{O}((dm/n)^p)$	$\mathcal{O}((dm/n)^q)$
LPC	$\mathcal{O}(n^{2-p}(dm)^q)$	$\mathcal{O}(n^{2-q}(dm)^q)$
LPC/BR	$\leq \mathcal{O}(m^{p-\hat{p}}d^pn^{1-p})$ $(1 < p < \hat{p})$	$d$ $(q=\hat{q}=1)$
	$\leq \mathcal{O}(d^pn^{1-p})$ $(1 < p=\hat{p})$	$\leq \mathcal{O}(m^{q-\hat{q}}d^qn^{1-q})$ $(0 < q < \hat{q} < 1)$
	$\leq d$ $(p=\hat{p}=1)$	$\leq \mathcal{O}(d^qn^{1-q})$ $(0 < q=\hat{q} < 1)$
		$\leq n/2$ $(q=\hat{q}=0)$

### 3.4.6.3 Super-linear Base Learner

Note that we have limited our focus in the training time analysis on the number of training examples, which is indeed only linearly proportional to the actual training time for base learners with linear complexity with respect to the number of examples. For some types of learning algorithms (such as decision trees, rule learners and also support vector machines) for which this relation is super-linear the pairwise approach may in fact benefit from distributing the work-load on a quadratic number of smaller problems.

In order to obtain a (rough) estimation of the behavior of these super-linear classifiers, we assume a class-balanced training set, i.e. each label shall have  $l/n = dm/n$  examples assigned. Consequently, we shall obtain subproblems with  $|\mathcal{T}_{rain}^{u,v}| \leq 2 \cdot l/n$  training exam-

ples for pairwise decomposition.<sup>30</sup> We proceed as in Section 3.1.1 and set the training costs for a problem with  $|\mathcal{T}_{rain}|$  to  $(|\mathcal{T}_{rain}|)^p$ .

Under these circumstances, the higher bound of the training complexity ascends to

$$\frac{n(n-1)}{2} \left( \frac{2dm}{n} \right)^p = O(n^{2-p}(dm)^p) \quad (3.28)$$

As it became clear, there is a trade-off between number of documents and label assignments  $l = dm$  which is fixed by the complexity grade  $p$ . Thus, the higher  $p$ , the more we can neglect the size of  $\mathcal{L}$  and the more we are dependent on the density  $l$  of the label matrix  $(\mathbf{y}_i)_i$ ,  $1 \leq i \leq m$ .

We remind the reader that the complexity of BR is  $nm^p$ , thus we obtain the following ratio between pairwise decomposition and BR

$$\frac{\frac{n(n-1)}{2} \left( \frac{2dm}{n} \right)^p}{nm^p} = \frac{n-1}{2} \left( \frac{2d}{n} \right)^p \lesssim d^p \left( \frac{2}{n} \right)^{p-1} \quad (3.29)$$

with the assumption  $n-1 \approx n$  for the interesting case of large  $n$ . We achieve runtime parity between the pairwise and comprehensive subproblems approaches (approximately) at  $\left( \frac{n}{2} \right)^{p-1} = d^p$ . Pairwise decomposition is hence more efficient as soon as

$$p > \frac{\log \frac{2}{n-1}}{\log(2\frac{d}{n})} \gtrsim \frac{\log \frac{2}{n}}{\log(2\frac{d}{n})} = \frac{\log 2 - \log n}{\log 2 - \log n + \log d} \quad (3.30)$$

For a learner with quadratic runtime, i.e.  $p = 2$ , both approaches are equivalent if the label density  $d/n$  is  $\frac{1}{\sqrt{2(n-1)}}$ , or  $d = \frac{n}{\sqrt{2(n-1)}} \gtrsim \sqrt{\frac{n}{2}}$ . Since the density of MLC problems is usually rather small, this point is quickly reached. From the opposite perspective, for the *rcv1* dataset e.g. this point would be already reached for a base learner with  $p = 1.42$ .

If we additionally assume that the simpler subproblems lead to more compact models, i.e. that  $p$  itself depends on the training set (cf. Section 3.1.1), we can compute an additional reduction. Let  $p$  be the exponent for learning a problem of size  $2 \cdot md/n$  and  $\hat{p}$  for learning a problem of size  $m$ , let  $p/\hat{p} < 1$ . Then the comparison between pairwise and the one-against-all binarization is (upper bound)

$$\frac{\frac{n(n-1)}{2} \left( 2d\frac{m}{n} \right)^p}{nm^{\hat{p}}} \lesssim m^{p-\hat{p}} d^p \left( \frac{2}{n} \right)^{p-1} \quad (3.31)$$

with  $n-1 \lesssim n$ . The more examples in the original problem, the less the label density and the more pronounced the ratio  $p/\hat{p}$ , the sooner we arrive at parity.

<sup>30</sup> The equality is only given for multiclass problems, i.e.  $|P_{\mathbf{x}}| = 1$  for all  $\mathbf{x}$ . The gap that results otherwise corresponds to examples that are part of both opposing classes, which can be a considerable number. Following Eq. 3.27 the sum of the differences  $\sum_{1 \leq u < v \leq n} (2 \cdot l/n - |\mathcal{T}_{rain}^{u,v}|)$  is  $\sum_{\mathbf{x}} |P_{\mathbf{x}}|^2 - |P_{\mathbf{x}}|$ .

As already mentioned in the present analysis, the fraction in Eq. 3.29 is in any cases at most  $d$  (for  $p = 1$ ). Note also that if calibration is used, this ratio can never be in favor of the pairwise approach since CLR always has to train additionally the one-versus-rest classifiers of BR.

While training of the base classifier can be linear or super-linear, the model and hence the testing complexity is usually constant or at most linear. We adopt the same assumptions as for the analysis of training and set the model complexity of a base model to  $\mathcal{O}(|\mathcal{T}_{rain}|^q)$  with  $q > 0$  (and usually  $q \leq 1$ ). Then the storage complexity, as well as the testing complexity, is similarly to Eq. 3.28 as follows

$$\frac{n(n-1)}{2} \left( \frac{2dm}{n} \right)^q \quad (3.32)$$

and the same ratios apply. Particularly, the ratio is bounded by  $\frac{n}{2}$  and  $d$  for  $0 \leq q \leq 1$ . Assuming a similar dependence of  $q$  on the size of the training set, the same relation as in Eq. 3.31 with  $q$  and  $\hat{q}$  instead of  $p$  and  $\hat{p}$  holds. They are provided in big  $\mathcal{O}$  notation and in tabular form together with the remaining estimations deduced in this section in Table 3.3.

Previous analyses in the literature have often assumed an equivalence between number of classifiers and total model size and in particular actual prediction time, and neglected the dependency on the number of training examples and other factors like problem hardness (cf. partially Fürnkranz 2002, Fürnkranz et al. 2008), though Rifkin and Klautau (2004) already casually referred to this point in context of support vectors and SVMs. In fact, Milgram et al. (2006) were clearly surprised by their experimental results. On a digit recognition task training the one-against-all SVMs (cf. Section 4.1.6) took 50 times more time, and on a 26 letters task 12 times longer. The BR model also contained 48% and 21% more support vectors, making the pairwise ensemble even faster in predicting.

### 3.5 Discussion

The following sections discuss learning by pairwise decomposition under several aspects, particularly in comparison to binary relevance decomposition. The decomposition itself and its direct effects are discussed in Sections 3.5.1, 3.5.2 and 3.5.3. The aggregation into label rankings (Section 3.5.5) using different voting strategies (Section 3.5.6) and the subsequent bipartitioning, particularly compared to calibration (Section 3.5.4), are points that are also thoroughly discussed. Section 3.5.7 and 3.5.8 extensively show the differences to the comprehensive subproblems generating decomposition schemes BR and ECOC. We conclude with a summary of the advantages and disadvantages of pairwise decomposition in Section 3.5.9.

#### 3.5.1 Easier Subproblems

The pairwise binarization method is often regarded as superior to binary relevance because it profits from simpler decision boundaries in the subproblems. In an early com-

parative study Fürnkranz (2002) confirmed the advantage of the pairwise decomposition technique on a set of multiclass problems from the UCI database (Frank and Asuncion 2010) for a rule learning algorithm as base classifier and Hsu and Lin (2002) found similar results for using SVMs. Knerr et al. (1992) observed that the classes of a digit recognition task were pairwise linearly separable (cf. Section 4.1.1), i.e. a function was found that perfectly discriminated between the examples of two digits, while the corresponding one-against-all tasks were not linearly separable.

One explanation for these improved results is the decomposition into smaller subproblems. In the case of an equal class distribution, the subproblems have  $\frac{2}{n}$ , and in the case of MLC,  $\frac{2d}{n}$  times the original size while binary relevance maintains the size (cf. Section 3.4.6.3). Typically, this goes hand in hand with the discriminating and generalizing power of a base learning. The discussion in Section 4.5.1 contains an illustrative explanation of this effect for a linear classifier.

### 3.5.2 Non-Competent Base Classifiers

We consider a classifier discriminating  $\lambda_u$  and  $\lambda_v$  as *non-competent* for a given instance  $\mathbf{x}_j$  if  $\lambda_u, \lambda_v \in P_j$  or  $\lambda_u, \lambda_v \in N_j$ . Recalling Figure 3.4 and Section 3.4.3, it may be disturbing at first sight that many non-competent base classifiers are involved in the voting process:  $h_{1,2}$  is asked though it cannot know anything relevant in order to determine if  $\mathbf{x}$  belongs to  $\lambda_1$  or  $\lambda_2$  since it was neither trained on this example nor on other examples belonging simultaneously to both classes  $\lambda_1$  and  $\lambda_2$  (or to none of both). In the worst case the resulting noisy votes (votes from non-competent base classifiers) concentrate on a single negative class, which could lead to misclassifications.

But note that any class can at most receive  $n - 1$  votes, so that in the extreme case when the competent BCs all classify correctly and the non-competent ones concentrate on a single class, a positive class would still receive at least  $n - |P|$  and a negative at most  $n - |P| - 1$  votes. Class  $\lambda_3$  in Figure 3.4 is an example for this case: It receives all possible noisy votes but still loses against the positive classes  $\lambda_1$  and  $\lambda_2$ .

Loza Mencía (2006, Section 5.6) continued the analysis and found that the expected margin  $v_u - v_v$ ,  $\lambda_u \in P, \lambda_v \in N$  between votes for the positive and negative labels is  $n(\frac{1}{2} - \text{ERR})$ , with ERR as the average error of a base classifier and assuming a non-biased prediction of the non-competent classifiers. Hence, the margin is independent of the number of  $|P|(n - |P|)$  competent and  $\binom{|P|}{2} + \binom{n-|P|}{2}$  non-competent BCs, i.e., interestingly, it is independent of the size of  $P$ .

### 3.5.3 Instances in the Label Intersections

Note that we explicitly do not consider or model the case where both labels are simultaneously present, i.e. a classifier  $h_{u,v}$  ignores instances with  $\lambda_u, \lambda_v \in P$ . The reason is that we consider multilabel learning from the perspective of *preference learning* (cf. Sec-



tion 2.5.4). I.e., we model and learn preference relations between labels which are strict and total, and hence no statement  $\lambda_u \succsim \lambda_v$  or  $\lambda_v \succsim \lambda_u$  can be given for the above case.

The advantages of this restriction are, firstly, the reduced complexity of the pairwise problems since we only have to learn one discrimination between  $\lambda_u$  and  $\lambda_v$ . Moreover, the problematic examples which lie in the intersection can be ignored (cf. Figure 3.2), further simplifying the learning of the separation. In fact, several extensions of the pairwise approach, such as pairwise correcting classifiers (Moreira and Mayoraz 1998) and tri-class SVMs (Angulo et al. 2006), integrate the remaining examples into the training process. The first one trains for each pairwise classifier  $h_{u,v}$  an additional two-vs-rest classifier  $h_{uv,r}$ , which discriminates examples of class  $\lambda_u$  and  $\lambda_v$  to all other classes. By assuming that  $h_{u,v}$  and  $h_{uv,r}$  correspond to the probabilities  $P(\mathbf{x} \in \lambda_u \mid \mathbf{x} \in \lambda_u \vee \lambda_v)$  and  $P(\mathbf{x} \in \lambda_u \vee \lambda_v)$  respectively, their product corresponds to the probability that testing instance  $\mathbf{x}$  belongs to one specific class. The prediction is then done by returning the class which has the highest probability. The latter one integrates the remaining examples by changing the original optimization problem of SVMs such that the remaining examples, roughly said, have to lie in the margin space. Petrovskiy (2006) learn SVMs that separate examples *only* in  $\lambda_u$ , i.e.  $y_u = 1 \wedge y_v = 0$ , from the remaining ( $y_v = 0$ ), and similarly SVMs for the opposite case *only*  $y_v$ . The predictions are interpreted as probabilities and combined following a pairwise probabilistic model in an expensive minorization-maximization optimization which is already unfeasible for  $n > 100$ . In several experiments these three approaches have lead to an improved performance, but which has to be paid with a considerable increase in training time, and more complex decision boundaries for the involved classifiers.

Secondly, it is not clear how the process can be made compatible to the current aggregation strategy. It would be necessary to develop a voting strategy that is semantically clean and consistent. E.g., if a classifier predicts both classes, distributing 0.5 votes to each would underestimate the statement, but distributing 1 to each would give this individual classifier more weight than others.

Certainly, one problematic case is that of hierarchical data if considered as a multilabel problem (cf. Section 2.5.6). Suppose that a class  $\lambda_u$  is the parent of  $\lambda_v$ , i.e.  $\lambda_u \succ \lambda_v$ . Then  $h_{u,v}$  would only receive positive training examples since there are no examples where  $y_u = 0 \wedge y_v = 1$ . There are several possibilities to deal with this case: firstly, the case  $\lambda_u$  vs.  $\lambda_v$  could be ignored. This would require to accordingly normalize the votings, since both classes will see their maximal number of votes diminished. Secondly, one could opt for predicting the a priori more probable class, namely  $\lambda_u$ . This would be semantically correct but of course would constitute a limitation, since  $\lambda_v$  would miss one vote. Note however, that in practice the problem generally disappears with increasing number of labels, since the remaining  $n - 2$  votes outweigh the missing vote. Moreover, note that the intersection respecting pairwise approaches mentioned above will face similar difficulties since there are no examples for  $\lambda_v$  and not  $\lambda_u$ .



---

### 3.5.4 Bipartitioning and Calibration

As was already mentioned in Section 3.4.4 and 2.5.5, multilabel classification is often reduced to predicting a ranking of labels or estimating the relevance of the individual labels. We are hence often confronted with a vector  $\mathbf{v} \in \mathbb{R}^{\mathcal{L}}$  of scores, probabilities, confidences, votes or similar (e.g. Eq. 3.6 for BR or Eq. 3.17 for LPC), which has to be translated into a subset of labels. A diverse set of solutions exist for this. However, we will focus in this section on methods that learn a function  $t$  that either returns an absolute threshold on  $v_i$ , which can then be used as in Eq. 3.4 or 3.25, or the number of top classes in a ranking that should be returned. Therefore, we will assume for the general case, and without loss of generality, a *bipartitioning* or *thresholding* function as in

$$t : \mathbb{R}^{a'} \rightarrow \mathbb{R} \quad (3.33)$$

with a number of input dimensions  $a'$ , which has yet to be determined. The function  $b$  in Eq. 3.21 hence either predicts  $\{\lambda_i \mid v_i \geq t\}$  or  $\{\lambda_i \mid r(\lambda_i) \geq t\}$ .

Given these preconditions, bipartitioning approaches can be categorized according to the following aspects. However, as we will see further below, some methods do not perfectly fit into this schema.

- **Input space:** For the input data on which the estimator will base its prediction, the original input vectors in  $\mathbb{R}^a$  can be maintained ( $a' = a$ ) or replaced ( $a' = n$ ) by the score vectors  $(v_1, \dots, v_n)$ , by sorted score vectors  $(v_{\pi_1}, \dots, v_{\pi_n})$ , or by the ranking positions  $(r(\lambda_1), \dots, r(\lambda_n))$ .
- **Target:** The algorithm usually either learns to predict the label cardinality or a threshold on the scores  $v_i$ .
- **Estimator:** For learning the target either regression (Witten and Frank 2005, Sec. 4.6) or (ordered) multiclass learners are possible. Static estimators always predict the same value.
- **Training:** If the original data is used as input, all necessary information is already available so that the estimator can be trained directly on the training data. If the scores are used, these are usually computed on the training set (which can lead to overfitting) or a separate validation set. In the latter case one may use cross validation if the estimators can be aggregated.

Yang (1999) discusses and evaluates a few straightforward approaches for determining a zero-point. The first one, *RCut*, uses a fixed cardinality for all examples (the mean on the train set). The second one, *SCut*, proposes to optimize score thresholds for each label on a separate validation set. The final one, *PCut*, is actually out of scope of Eq. 3.33 since it is applied on the whole test set and predicts proportionally the same amount of relevant instances for each class as in the training set. Note that particularly *RCut* should not be considered a practicable approach, but rather an evaluation method (cf. Section 2.7.4).

One of the earliest approaches that adjust the thresholds for each individual example is described by [Elisseff and Weston \(2001\)](#). It learns a linear  $t : \mathbb{R}^n \rightarrow \mathbb{R}$ , which predicts a threshold on the scores, by the method of least-squares. [Petrovskiy and Glazkova \(2006\)](#) analyze a technique that computes a linear combination of the ranks which returns the cardinality and a more sophisticated method which additionally scales and translates the scores. A similar approach was presented by [Fan and Lin \(2007\)](#), which determines new thresholds for each underlying SVM individually using a validation set. The thresholds can be adapted according to a particular metric. Depending on whether micro or macro-averaged F1 should be optimized, several iterations of the greedy algorithm are required. The baseline is improved with respect to the selected measure. However, several additional techniques are necessary in order to restrict the risk of extreme distributions of the scores. [Tang et al. \(2009\)](#) extend the first approach of [Petrovskiy and Glazkova](#) and additionally try out scores and sorted scores as input, concluding that using  $\mathcal{X}$  performs best. [Ioannou et al. \(2010\)](#) provide an extensive study on the usage of RCut, SCut, labelset size and threshold predictors for different rankers such as ML-KNN, BP-MLL (cf. Section 2.8.2), BR and CLR.

The last study specifically compares the virtual label of CLR to 25 thresholding variants and it is shown that calibration is, admittedly, improvable.<sup>31</sup> Though the approaches were compared with respect to the problematic Hamming loss (cf. Section 2.7.3), CLR faced some difficulties regarding the correct prediction of the cardinality in our evaluations throughout this work, too. Section 4.6.9 will discuss possible reasons.

Nevertheless, calibration perfectly fits into the framework of pairwise preference learning and hence provides a natural extension to LPC in order to determine a zero-point for the predicted label rankings, which additionally directly depends on the underlying test instance. Note that calibration is not restricted to pairwise decomposition, but can be elegantly integrated in all label ranking methods that make use of pairwise preferences between labels as training signal. Furthermore, it is recommendable due to its simplicity also regarding the prerequisites: the base classifiers are not expected to produce accurate probabilities or scores. On the other hand we can certainly expect that more precise confidences improve the bipartitioning performance since better scores come hand in hand with better binary predictions. Moreover, though some efficient approaches are conceivable, it is generally more expensive to produce training data on validation sets for learning a threshold function than training and applying the additional  $\mathcal{L}$  binary classifiers.

### 3.5.5 Label Ranking

We already elaborated on the advantages of predicting label rankings in multiclass and multilabel classification in Section 2.5.4: the vast availability of soft classifiers, a clear

<sup>31</sup> Interestingly, the best approach estimates a fixed threshold for all  $v_i$  on the training set, which could indicate that relevant labels receive a certain minimum number of votes regardless of the actual varying labelset sizes.

**Table 3.4:** Example of joint and marginal probabilities for a label set  $\{y_1, y_2, y_3\}$  given a particular example  $x$ . The right column indicates the joint probability for the labelsets given on the left columns. The last row indicates the marginal probabilities of the labels. The table on the right indicates the probability given  $\lambda_1$  is positive.

$y_1$	$y_2$	$y_3$	$P(y = (y_1, y_2, y_3))$	$y_1$	$y_3$	$P(y_1, y_3   y_2 = 1)$
0	0	0	.05	0	0	.15
0	0	1	.05	0	1	<b>.38</b>
0	1	0	.10	1	0	.31
0	1	1	<b>.25</b>	1	1	.15
1	0	0	.15	.46	.54	$P(y_i = 1   y_2 = 1)$
1	0	1	.10			
1	1	0	.20			
1	1	1	.10			
.55	<b>.65</b>	.50	$P(y_i = 1)$			

interpretation of rankings and the exploitation of the more informative predictions, e.g. for a more exhaustive evaluation.

Label ranking, as described in Section 3.4.2 and 3.5.4, constitutes a restriction for multilabel setting since it is not possible to enumerate all possible labelsets with the help of a simple threshold. If the labels are sorted according to their relevance, or in other words, according to their marginal probability, it is generally not always possible to predict the true labelset, even if all positions in the ranking are evaluated as possible cut-off points. Consider the example given in Table 3.4: if the label were ordered according to their marginals (last row), the sequence would be  $\lambda_2 \succcurlyeq \lambda_1 \succcurlyeq \lambda_3$ . It would hence not be possible to predict the correct labelset  $\{\lambda_2, \lambda_3\}$ , which certainly would optimize every loss.

On the other hand this does not necessarily constitute a conceptual restriction. If a classifier is certain about a particular labelset, it can just predict it at the top of the ranking and set the predicted cut-off correspondingly. Hence, learners that aim at optimizing the joint distribution are not restricted in their expressiveness by the label ranking framework.

Furthermore, even if a ranker determines the output ranking according to the marginal probabilities, this does not generally exclude a perfect prediction. More complex bipartitioning approaches, that work as in the more abstract definition in Eq. 3.21, are able to manipulate the predicted ranking. A simple example was given in Section 3.5.4: the technique of Petrovskiy and Glazkova (2006) can add a bias to each label score and hence do a reorganization of the labels. The more sophisticated approach of Park and Fürnkranz (2008) models unconditional label dependencies on the training data in form of association rules on the output space. This could already suffice in our example if there was a general dependence  $\lambda_2 \succcurlyeq \lambda_3$  as e.g. in a hierarchical setting. Otherwise, the framework of Park and Fürnkranz is easily extensible in order to define such constraints on the local vicinity of a test example (cf. also the induction of local patterns in Chapter 9).

A different approach specific to the pairwise setting (regarding the costs) is described by Hüllermeier and Fürnkranz (2007). The authors describe an aggregation algorithm called *ranking through iterated choice* (RIC) based on *empirical conditioning* that works as follows: First, the conventional voting process is performed and we obtain a voting vector  $\mathbf{v}^{(1)}$ . The top class  $\lambda_{\pi_1}^{(1)}$  is then removed and the voting process is repeated as if  $\lambda_{\pi_1}^{(1)}$  did not exist. Again, we receive a vector  $\mathbf{v}^{(2)}$  (which can actually be efficiently computed from  $\mathbf{v}^{(1)}$  in the pairwise setting) and put  $\lambda_{\pi_1}^{(2)}$  at the second position after  $\lambda_{\pi_1}^{(1)}$ . This continues until the last label  $\lambda_{\pi_1}^{(n)}$  is determined.

RIC was initially developed in the context of multiclass classification in order to produce an accurate alternative if the predicted class was not correct, i.e. the classifier  $h^{(i)}$  in the  $i$ -th iteration approximates  $P(\lambda_j | \lambda_{\pi_1}^{(1)}, \lambda_{\pi_1}^{(2)} \dots \notin P, \mathbf{x})$  for each  $\lambda_j$  not already selected as the top class. It does not take much effort to re-interpret  $h^{(i)}$  as  $P(\lambda_j | \lambda_{\pi_1}^{(1)}, \lambda_{\pi_1}^{(2)} \dots \in P, \mathbf{x})$  in the multilabel setting (for which  $|P| = 1$  does not hold), i.e., as the probability of  $\lambda_j$  given that  $\lambda_{\pi_1}^{(1)}$  was already selected and is correct. Under this assumption, the product rule of probabilities in Eq. 2.28 and similarly to the probabilistic classifier chains described in more detail in Section 2.8.5, it is the intuitive notion of the author that RIC approximates the Bayes optimal ranking of labels. Certainly, applied to our example in Table 3.4, RIC would expectably choose  $\lambda_3$  assuming  $P(\lambda_1, \lambda_3 | \lambda_2, \mathbf{x})$  is correctly estimated by  $h^{(2)}(\mathbf{x})$ . We leave this idea for further research.

I would like to make a note on the criticism of Dembczyński et al. (2010c) that, roughly speaking, it is possible to optimize labelset-independent losses such as Hamming loss, but also the ranking loss (Kotłowski et al. 2011), by just finding a classifier that correctly predicts the marginal probabilities, and that it is therefore not necessary to take label dependencies into account if these losses are the objective. Without deeply analyzing this point, we can make the following observation on our toy example: A classifier that perfectly predicts the marginals (last row, risk minimizer in terms of Dembczyński et al.) would neither minimize Hamming loss nor ranking loss, but this would happen for a joint distribution optimized classifier. The experimental evaluation in Section 9.6.3 is not fully conclusive in this respect, also because Hamming loss is not considered, but it indicates that independently trained BR classifiers produce rather bad rankings while the label powerset classifiers focused on joint distributions perform quite well. We leave this as an open question and refer to the works of the referred authors for further reading.

In conclusion, despite the arisen points, we see label ranking in general and ranking by pairwise comparison in particular as a suitable framework for solving and modeling multilabel tasks. The shortcomings in flexibility are outweighed by the gained simplicity, and several approaches were presented in order to tackle most of the concerns regarding the limited expressiveness. As the next section shows, the aggregation strategy in the pairwise framework provides reliable rankings, which is the basis for a “good” multilabel classification.

---

### 3.5.6 Aggregation and Voting

Many different approaches were developed in order to combine the pairwise predictions into one overall result. [Wu et al. \(2004\)](#) analyzed different approaches under the aspect of accurate probability estimations, while [Galar et al. \(2011\)](#) provide a thorough comparison between a long list of aggregation schemes for pairwise and also binary relevance decomposition. Even the quicksort algorithm was adopted in order to produce rankings from pairwise comparisons ([Ailon and Mohri 2008](#)). Most of these methods were developed for being used in pairwise multiclass classification, however the results are usually transferrable to the multilabel case since there is no difference in the aggregation.

Nonetheless, the preferred method throughout this work is simple voting (cf. Section 3.4.3), though the presented learners do not exclude other schemes. We will motivate this in the following based on empirical as well as theoretical evidence.

First of all, simple voting can be used with any base classifier, since it only needs a binary prediction. Other approaches depend on soft or probabilistic classifiers that provide a score. Moreover, these estimates can usually not be adopted without further preprocessing. The outputs of (non-linear) SVMs e.g. usually lie piled around 1 or  $-1$  (cf. [Platt 2000](#), due to Eq. 4.6) and require an additional step of fitting a "normalizing" and smoothing sigmoid function ([Lin and Lin 2003](#), [Platt 2000](#)). Similar difficulties can be observed for rule learners, decision trees and even Naive Bayes. Simple voting in contrast is not committed to accurate probabilistic or confidence estimations.

Furthermore, it is known from ensemble techniques that simple voting provides theoretical guarantees of returning the correct decision. [Kuncheva \(2004, Sec. 4.2.1\)](#) shows that the accuracy of a voting process approximates 1 with increasing number of voters if the error of the base deciders is greater than 0.5. Similarly, previous work of [Loza Mencía \(2006, Sec. 5.6\)](#) shows (under the assumption of unbiased incompetent base classifiers, cf. Section 3.5.2) that the average margin in votes between relevant and irrelevant classes increases linearly with the number of labels and is further only dependent on the error rate of the base classifier.

Theoretical guarantees can also be given for the other simple approach: weighted voting (WV). WV is shown to minimize the Spearman rank correlation ([Hüllermeier et al. 2008](#)), which is a distance between rankings that sums the squared rank differences. Furthermore, weighted voting approximates the *optimal* adaptive voting prediction (AV), which is claimed to provide a so called maximum a posteriori (MAP) probability prediction ([Hüllermeier and Vanderlooy 2010](#)). This strategy assumes a certain distribution over the scores  $h'_{u,v}(\mathbf{x})$ , similarly to sigmoid fitting (cf. above), and chooses the parameters  $\alpha_{u,v}$  separately for each classifier in order to fit the distribution. The voting function  $a : [0; 1] \rightarrow (-\infty; 0]$  then produces the (static) adapted scores  $-\log(1 + \exp(\alpha_{u,v}(1 - h'_{u,v}(\mathbf{x}))))$ .<sup>32</sup> Strong, accurate classifiers (more clear scores near 0

---

<sup>32</sup> Interestingly, using the aggregation scheme of [Price et al. \(1994\)](#), which sums up the reciprocals of the predicted scores as penalties, results in a very similar equation if sigmoid fitting is applied to the classifiers. The idea behind this scheme is to penalize classes which obtain clear votes against. In the comparative study of [Wu et al. \(2004\)](#) this method showed very stable results though it was compared

---

and 1) receive higher  $\alpha$ 's, while the amplitude of the adapted scores for weak predictors is reduced.

Under certain assumptions which are easily (or often) satisfied, [Hüllermeier and Vanderlooy](#) prove that WV approximates the MAP prediction and hence produces equal or similar label rankings than AV. However, WV makes less assumptions and is thus expected to be more robust than AV if the underlying model is not met. The similar power of WV is confirmed in an experimental evaluation on a set of multiclass problems ([Frank and Asuncion 2010](#)).

These experiments also demonstrate that simple voting performs similarly at the same degree. Particularly, it is observed that simple voting performs less good in predicting the top class (with one clear loss against AV and WV for three different base learners), but on the other hand much better (one clear win) on position error (comparable to MARGIN, Eq. 2.50). This is to some extent in line with the results of [Wu et al. \(2004\)](#), which saw simple voting as a poor probability estimator, but a good ranker. A revision of the results of [Galar et al. \(2011, Table 4\)](#) further reveal that LVPC, WV and simple voting (in that order) achieve the highest mean and median of the 6 average ranks given for the 6 base classifiers used and comparing 8 different aggregation strategies on 19 UCI datasets. Learning valued preference for classification ([Hühn and Hüllermeier 2009](#), [Hüllermeier and Brinker 2008](#)) decomposes a pairwise prediction  $r_{u,v} = h'(\mathbf{x})$  into the three fuzzy components strict preference, conflict and ignorance, which are then combined into a weighted vote and which [Galar et al.](#) compute, effectively, as  $v_{u,v} = r_{u,v} + (m_u/m - 1/2) \cdot \min(r_{u,v}, 1 - r_{u,v})$  with  $m_u$  as the number of positive examples for training  $h'_{u,v}$  and  $m$  as the total number of examples in the subproblem.

In conclusion, several studies show a, overcautiously expressed, comparable performance of simple voting with the other simplistic scheme, weighted voting, as direct competitor. Since simple voting neither requires soft-classifiers nor any pre-adaptation of the scores, we opine that using this voting strategy constitutes a very reasonable and arguable decision.

### 3.5.7 Comparison to Binary Relevance Decomposition

The discussion about whether one-against-all (OAA) or the pairwise approach is preferable is very old and originates from the fact that pairwise decomposition was the first *other* binarization technique beside OAA, before ECOCs in the year 1995 and of course long before label powerset, which came up with multilabel classification. As the only direct competitor, pairwise decomposition was by default compared to OAA. In the next section, we review some of the main works comparing OAA and the pairwise approach and some of the most debatable points and criticisms.

---

to more sophisticated approaches. However, both approaches bear the risk that the distribution of small scores is underestimated by the chosen distribution model, and consequently that small scores are over-penalized. As the authors of AV point out, their assumption that the scores are distributed according to a truncated exponential distribution does often not hold in practice.



The first work in the field of machine learning which employed pairwise decomposition that is known to the authors was in 1990 by [Knerr et al.](#), who continued their effort in some other early works. They observed e.g. that the classes of a digit recognition task were pairwise linearly separable, while the corresponding one-against-all task was not solvable with their linear neural network. Several other works followed, which mainly showed the same picture but were not dedicated to binarization techniques (cf. references in [Fürnkranz 2002](#)). These were considered only as an additional tunable configuration setting. The first methodological studies were presented by [Hsu and Lin \(2002\)](#) and [Fürnkranz \(2002\)](#). The first work compared OAA with conventional pairwise decomposition and an alternative aggregation strategy specific to multiclass problems. LPC consistently outperformed OAA. [Fürnkranz](#) presented the first formal analysis of pairwise decomposition and showed that, surprisingly, training a quadratic number of base classifiers was more efficient than training the linear number of OAA models.

In a direct response to this study, [Rifkin and Klautau \(2004\)](#) firmly argued in favor of OAA. Their main claim was that if the base learners, in their case support vector machines, were appropriately tuned, then there should be no advantage for neither approach. Their intuition is that when SVMs are tuned ( $C$  and kernel parameter, see also Section 4.1.6), only mistakes are made for examples that “simply for all practical purposes *look* more like a member of an incorrect class”, and that to correctly classify this type of examples is very difficult, for any decomposition approach. [Rifkin and Klautau](#) themselves deliver the cases where LPC expectedly achieves an advantage. Firstly, optimizing SVM parameters is very costly. Indeed, a greedy approach is used in order to find good global parameters, in contrast to the recommended grid search (cf. [Chang and Lin 2001](#)) on every subproblem separately. Secondly, the authors expect that using weak or improperly tuned base learners will have an adverse impact on OAA. Their second point can be confirmed to a certain extent later in the present thesis, where the fast but also simple perceptron algorithm is shown to be a backbone for the efficient pairwise multilabel classification of large data. Very recently, [Galar et al. \(2011\)](#) presented an extensive study comparing OAA and a dynamically ordered version (see [Fürnkranz 2002](#), for statically ordered OAA) to nine different pairwise aggregation strategies. The pairwise approaches clearly outperform OAA for six different base learner. However, none of the base learner was tuned and two were certainly weak ( $k$ -NN with  $k = 1$  and 3).

The perhaps most important statement in the work of [Rifkin and Klautau \(2004\)](#) is however, that their main claim is explicitly only valid if the classes are independent, which is certainly true for multiclass data but rarely for multilabel data. As a consequence, many studies in the frame of multilabel classification comparing pairwise decomposition favorably to binary relevance appeared, including most of those cited in Section 2.8 (BR is, again, almost always used as a baseline) and including the studies about pairwise classification brought together and presented in this thesis.

However, recently, two new works appeared in defense of BR ([Read et al. 2009, 2011](#)). The proposed method of classifier chains (CC) fixes a particular order of the BR base classifiers and subsequently adds the outputs of the preceding classifiers as new features. Hence, the  $i$ -th classifier  $h_i : \mathcal{X} \times \mathcal{Y}^{i-1} \rightarrow \mathcal{Y}^1$ , with  $\mathcal{Y} = \{0, 1\}$ , is trained on  $j = 1 \dots m$  ex-

amples  $((\mathbf{x}_j, y_{j,1}, \dots, y_{j,i-1}), y_{j,i})$  and predicts on a test instance  $(\mathbf{x}, h_1(\mathbf{x}), \dots, h_{i-1}(\mathbf{x}, \dots))$ . The objective of such an approach is clear: to tackle the main drawback of BR, the ignorance of the dependencies between labels (as stated by Rifkin and Klautau). The proposed method clearly satisfies this multilabel-specific demand. It is shown by Dembczyński et al. (2010a) that CC takes conditional dependencies into account (cf. Section 2.6) and deterministically approximates the optimal Bayes classifier.

But it is the firm opinion of the author that CC cannot be submitted as an argument in defense of the BR approach since it is conceptually not equivalent to BR. The chaining part has to be considered as a novel and intelligent method of *stacking* which is used *on top* of BR. But stacking itself is not restricted to BR or any other decomposition approach since it simply relies on extending or replacing the input space (cf. Section 2.8.5) and was already extensively used in the context of multilabel learning (cf. Section 9.7). In the eyes of the author, the main contribution of CC is hence this novel, intelligent and very interesting stacking approach. In conclusion, stacking can not be used as an argument for a systemic or conceptual advantage of BR, just as the stacked variants of pairwise classifiers recently introduced by Madjarov et al. (2012) cannot be instanced in favor of a general superiority of pairwise decomposition.

The steady argumentation in favor of BR is due to the emphasized beneficial scalability properties of BR, since BR (and also CC) scale linearly with the number of classes (cf. Section 3.1.1). However, as we have clearly seen in the comparison in Section 3.4.6, LPC is comparable or under certain circumstances even faster than BR in training, which is the phase mainly considered by Read et al.. Surprisingly and for unknown reasons, CLR does not terminate on relatively small text datasets with 22 and 29 classes in their experiments, whereas we were able to apply similar SVMs as they used them on datasets with up to 159 classes without any problems. Moreover, Read et al. use an ensemble of CCs for their comparison to other approaches, which further decreases efficiency and scalability by a predetermined factor, usually by 10 or 50 times. An additional note is that with increasing number of classes (to an order of magnitude of 1000s) the positive effect of the stacking vanishes and ensemble CC even drops below BR for some losses.

In summary, despite the occasional criticisms, we see strong arguments in favor of pairwise learning compared to binary relevance decomposition. Pairwise learning has shown to dominate BR in a large number of experimental studies on multiclass problems, and more recently, on multilabel problems. Part of this empirical evidence is provided by the studies of the author and reflected in this thesis. However, also in this work, we see circumstances where BR-based approaches can be more advisable, particularly regarding highest scalability and highest efficiency demands. The MMP algorithm is a good example for this (cf. Chapter 4), although, performing certain simplifications, pairwise classification is applicable even under these circumstances (cf. Chapter 7).

### 3.5.8 Comparison to Ternary ECOC

Note that pairwise classification can be seen as a special case of the generalized ternary Error Correcting Output Codes (ECOC) framework (Allwein et al. 2000) with a fixed en-



coding matrix. Similarly to Section 3.3, the decoding matrix for the example in Figure 3.2 would be

$$\begin{matrix} & h_{1,2} & h_{1,3} & h_{2,3} \\ \lambda_1 & \left( \begin{array}{ccc} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & -1 \end{array} \right) \end{matrix} \quad (3.34)$$

We believe that the pairwise scheme is a case that deserves special attention for several reasons. First and foremost, it has clearly defined semantics. Each binary classifier determines which of two labels is to be preferred for a given example. This is the smallest piece of information that is needed for establishing an order between all labels. Second, it is a fixed, domain-independent and non-stochastic decomposition method that has a good overall performance. In several experimental studies (including Allwein et al. 2000), it performed on par or better than competing decoding matrices. The chief reason for the good performance is that two-class problems are often linearly separable, even in low-dimensional spaces (cf. Section 3.5.1). Finally, it is also among the most efficient decoding schemes.

In some sense, our philosophy is orthogonal to ECOC: While ECOC puts its efforts into choosing a good encoding matrix, we fix the encoding to the all-pairs approach and concentrate on the decoding phase. We believe that many practical problems can be reduced to estimating pairwise probabilities. These can then be combined in various ways, optimizing different performance criteria with a single, fixed pairwise ensemble (cf. Section 3.5.6).

### 3.5.9 Summary: Advantages and Disadvantages

The following listing summarizes the multiple advantages and disadvantages of learning by pairwise comparison (LPC) in the frame of multilabel classification, particularly in comparison to binary relevance (BR). They can mainly be categorized as systemic (system inherent) versus empirical arguments, though it is the objective of this work to provide the experimental evidence in the following chapters. Some obvious points are introduced for the first time in this list.

- LPC is comparable to BR in training complexity within a small factor, which is determined by the average label cardinality in the training data (cf. Section 3.4.6). With a super-linear base learner, the comparison can even turn out to be favorable for LPC (cf. Section 3.4.6.3)
- LPC has to train and evaluate a quadratic number of base classifiers (cf. Section 3.4.6) and maintain them in memory. It is the main objective of this work to analyze and develop solutions in order to overcome this bottleneck.

- 
- LPC allows a high degree of parallelization, which becomes even more important with increasing number of cores per processor and advantages in distributed computing *in the cloud*. In contrast to BR, the number of possible parallel jobs is incremented by a factor linear in the number of classes.
  - LPC allows adding or removing classes from the model even after training. It allows *class-incremental* training and testing.
  - LPC decomposition produces smaller problems which are easier to learn, with respect to accuracy as well as time (cf. Section 3.5.1).
  - LPC and the chosen model of strict total preferences only considers the case of pairwise exclusion and ignores pairwise co-occurrence of labels, which could lead to the loss of valuable information (cf. Section 3.5.3).
  - Basic LPC is parameter-free. No configuration and costly parameter fitting is necessary. Moreover, no prerequisites are made to the underlying base learner, which also makes the base learner easier to configure.
  - The extension of calibration naturally integrates into framework of pairwise preference learning. It provides an elegant solution to bipartitioning and also works absolutely parameter-free *out of the box* (cf. Section 3.4.5, 3.5.6).
  - Within the general framework of preference learning, LPC allows to encode and model the smallest piece of information available, namely pairwise relations between classes (cf. Section 3.5.8).
  - LPC training does not allow for direct minimization of a particular measure such as e.g. holistic approaches (cf. Section 2.8.2). However this step is transferred to the aggregation on the basis of instance-dependent pairwise preferences, for which several specialized solutions exist (cf. Section 3.5.6). Moreover, in contrast to holistic approaches also considering pairwise preferences between labels in a global optimization problem, such as Rank-SVM (Elisseeff and Weston 2001) or SVM<sup>rank</sup> (Joachims 2002, 2006), LPC is much more efficient in training.
  - And finally, LPC has shown to be superior to BR in a wide range of studies on multiclass and multilabel classification with respect to predictive quality (cf. Section 3.5.7).

The following chapters will particularly focus on the first two points in the list regarding the efficiency of pairwise decomposition and present appropriate solutions. However, the remaining points will accompany us through the whole work and new aspects will be subsequently added.

---

## 4 Pairwise Learning of Efficient Perceptrons

A main challenge for multilabel classification in general and for pairwise classification in particular is the amount of instances that have to be processed (cf. Section 1.1). This chapter introduces several possible solutions, all based on the usage of an efficient base learner. The examination is not restricted on the solution of this particular challenge. This study also allows to experimentally evaluate the most relevant approaches in Chapter 3, namely binary relevance and pairwise decomposition and the associated calibration technique, and to empirically confirm some of the advantages and disadvantages claimed in Section 3.5.

Crammer and Singer (2003) combined the one-against-all (cf. Section 3.1) method and the label ranking (cf. Section 2.5.4) idea in their multiclass multilabel perceptron algorithm (MMP). Instead of learning the relevance of each class individually and independently, MMP incrementally trains the entire classifier ensemble as a whole so that it predicts a real-valued relevance score for each class. This is done by always evaluating the performance of the entire ensemble, and only producing training examples for the individual classifiers when their corresponding classes are incorrectly ordered in the ranking.

Perceptrons are used as base classifiers. This algorithm has recently received increased attention, especially in text classification, since it is simple, efficient and effective. In addition, perceptrons allow incremental training, which makes them particularly well-suited for large-scale classification problems such as the large Reuters Corpus *rcv1* benchmark (cf. Section 2.9.1).

It is composed of more than 800,000 documents, which are assigned to on average 3.2 of 101 possible classes. This collection constituted a new challenge in text classification and in particular in multilabel classification. It is still one of the datasets with the largest amount of documents, one of the key dimensions of MLC scalability (cf. Section 1.1). The *rcv1* corpus is also an early representative for similar collections from the Web 2.0, which came up to an increased extent in the years following the publication of *rcv1* in 2004.

In this chapter, we propose the use of pairwise decomposition as an alternative training method for an effective and efficient ensemble of perceptrons. Multilabel pairwise perceptrons (MLPP) are trained and used as described in Section 3.4, i.e. we train one classifier for each possible class pair and we test by producing one overall label ranking by combining the predictions of the individual classifiers by simple voting.

This first study demonstrates the multiple advantages of learning and classifying by pairwise comparison, as well as using the fast perceptrons as base learner: Despite the quadratic number of perceptrons and the additional information between labels pro-

cessed, MLPP's training is competitive to BR's and MMP's since its costs only differs by a constant factor. In addition, the work of [Loza Mencía \(2006\)](#), followed by [Loza Mencía and Fürnkranz \(2007b, 2008c\)](#), was one of the first works confirming the superiority of the pairwise approach on the multilabel and label ranking task. The main reason for this is that, while BR and MMP propose to include information about the ranking task into the training signals, the pairwise approach addresses the ranking and bipartition problem by breaking the ranking signal down into elementary binary preferences that induce a final ranking (cf. Section 3.4.2). As it turns out, perceptrons seem to particularly benefit from the smaller and thus simpler pairwise subproblems.

The basic version of MLPP described herein still evaluates a quadratic number of perceptrons for prediction, and we will see in Chapter 5 how to substantially improve this circumstance. However, the study in this chapter still demonstrates that pairwise perceptrons are very suitable for the demands and challenges that are imposed by large datasets with a vast amount of documents such as the Reuters *rcv1*. At the time of development, it was also the most exhaustive evaluation of pairwise classification in terms of label dimensionality, and probably also in terms of number of features. To the best of our knowledge, it is also the first and unique study dedicated to incremental pairwise learning.

Furthermore, we extended MLPPs by the calibration technique introduced in Section 3.4.5. This chapter reflects the examination on four dataset from different domains carried out by the first major study<sup>33</sup> on this extension of the pairwise preference learning framework ([Fürnkranz et al. 2008](#)). The study demonstrated the effectivity of the artificial label and combination with the conventional relevance classification approach in order to produce accurate bipartitions based on the label ranking outputs. Although we did not evaluate competing thresholding techniques (cf. Section 3.5.4), the experimental framework chosen allows us to evaluate independently from bipartitioning. Hence, we are still able to deduce that even using highly accurate thresholding techniques, the calibrated pairwise approach would outperform the BR approaches.

This chapter is organized as follows: Firstly, we introduce the base learning algorithm, the perceptron, in Section 4.1. Sections 4.2 and 4.3 continue with the description of the competing multilabel algorithms, the MLPP algorithm itself is discussed in Section 4.4. While Section 4.5.2 provides an extensive analytical comparison, Section 4.6 evaluates the approaches experimentally. Section 4.6.9 discusses the results and Section 4.7 summarizes the study in this chapter.

## 4.1 Perceptrons

A perceptron is a binary classifier initially developed as a model of the biological neuron ([Rosenblatt 1958, 1988](#)). Internally, it computes a linear combination of a real-valued input vector and predicts the positive class if the result is positive, and the negative class

<sup>33</sup> [Brinker et al. \(2006\)](#) firstly introduced the calibration technique, but their evaluation was only on a very small subset of *rcv1* and on only one additional dataset. In addition, this work focused on the comparison of pairwise and uni-label-focused decomposition, while the present chapter takes actual improvements and adaptations to label ranking into account.

otherwise. Therefore, it belongs to the family of linear classifiers, which is defined in the following.

#### 4.1.1 Linear Classifiers

A linear classifier consists of a weight vector  $\mathbf{w} \in \mathcal{X} = \mathbb{R}^a$  and a threshold  $b \in \mathbb{R}$  and is given by the following classifier function for an input instance  $\mathbf{x}$ :

$$h'(\mathbf{x}) := \mathbf{x} \cdot \mathbf{w} + b \quad h(\mathbf{x}) = I[h'(\mathbf{x}) > 0] \quad (4.1)$$

with the indicator function  $I$  (cf. also Eq. 3.4).

We can interpret a linear classifier as a hyperplane with the formula  $\mathbf{x} \cdot \mathbf{w} = -b$  that divides the  $a$ -dimensional space into two halves. An instance is a point in this space and its position determines whether it is considered as belonging to one class or the other. If the two sets of positive and negative points, respectively, can be separated by a hyperplane, they are called *linearly separable*. As a consequence, irrespective of the training algorithm used, linear classifiers such as the perceptron cannot arrive at correct predictions for all potential instances unless the negative and positive instances are linearly separable.

#### 4.1.2 Perceptron Training Rule

In order to find a possibly existing *separating hyperplane*, the perceptron algorithm adapts the weights after each training example  $\mathbf{x}_i$  with the binary class  $y_i \in \{0, 1\}$  according to the following training rule:

$$\begin{aligned} \alpha_i &= (y_i - h_i(\mathbf{x}_i)) \\ \mathbf{w}_{i+1} &= \mathbf{w}_i + \alpha_i \mathbf{x}_i \\ b_{i+1} &= b_i + \eta \alpha_i \delta \end{aligned} \quad (4.2)$$

with  $h_i(\mathbf{x}) := I[\mathbf{w}_i \mathbf{x} + b_i > 0]$ .<sup>34</sup> Hence,  $\alpha_i$  indicates if there was a classification error on the current example ( $\alpha_i \neq 0$ ), and if, and only if, this was the case, the model is updated by adding or subtracting the training example to the weight vector depending on the sign of the error (the bias is updated accordingly). This moves the hyperplane towards the misclassified point.

The bias term  $\delta$  is usually being set to 1 (Section 4.1.3) and the initial weights  $\mathbf{w}_0, b_0$  are set to zero without loss of generality. The learning rate  $\eta$  can be ignored if set to be constant (Bishop 1995), as it will be usually the case in this work. Note also that this

<sup>34</sup> Linear classifiers are per definition symmetric, i.e. a classifier  $h'^{-}$  predicting the negative instead of the positive class is simply given by  $-h'(\mathbf{x}) = -\mathbf{x}\mathbf{w} - b$  (cf. Eq. 4.1). However, neither the test  $h'(\mathbf{x}) > 0$  nor  $h'(\mathbf{x}) \geq 0$  completely suffice the symmetry, since  $h(\mathbf{x}) + h^-(\mathbf{x}) \neq 1$  for  $h'(\mathbf{x}) = 0$ . In theory,  $> 0$  leads to a more conservative strategy that enforces perceptrons to undo such an inconsistent state. In practice however,  $h'(\mathbf{x}) = 0$  is very improbable due to the random initialization with continuous values of  $\mathbf{w}$  and  $\geq 0$  can be safely used in implementations, as we did.

implicitly corresponds to subsequently decreasing the learning rate, since with increasing  $i$  (or more specifically, with increasing number of errors  $\alpha_i \neq 0$ ) the norm of the vector  $b_{i+1}$  increases so that the impact or influence on the model of a correction  $\alpha_i \mathbf{x}_i$  decreases (see also Section 4.6.7 for a comparison).

As can be clearly seen from Eq. 4.2, perceptrons are *incrementally trainable*, or in other words, the perceptron algorithm is an *online-learner*: the model is immediately updated after each training example in the training sequence and immediately ready for being used for a prediction (cf. [Sebastiani 2002](#), Sec. 6.6). Hence it is additionally an *anytime classifier*. In fact, the perceptron algorithm begins with the classification of the current training example in the first line of Eq. 4.2. The perceptron can be trained in several epochs, i.e. after the last example  $\mathbf{x}_m$  in  $\mathcal{T}_{rain}$  was seen in the first epoch, the training continues with  $\mathbf{x}_{m+1} = \mathbf{x}_1$  until  $Tm$  training examples were used in total for training in  $T$  epochs.

If the training examples are seen iteratively in this way and the data is linearly separable, the algorithm provably finds a dividing hyperplane (in a finite number of epochs). This is called the perceptron convergence criterion (cf., e.g., [Bishop 1995](#)). Irrespective of training until convergence not always being desirable, this property does not reveal anything about the performance on unseen data.

#### 4.1.3 Bias and Threshold

Certainly, the  $\delta$  value becomes important when the perceptron is trained in only one epoch: it is easily shown that  $|\mathbf{w}_{m+1}| \leq |\mathcal{W}| \cdot \max_{\mathbf{x} \in \mathcal{W}} |\mathbf{x}|$  and  $|b_{m+1}| \leq |\mathcal{W}| \cdot \delta$  holds for misclassified training examples  $\mathcal{W} = \{\mathbf{x}_i \in \mathcal{T}_{rain} \mid h_i(\mathbf{x}_i) \neq \mathbf{y}\}$ . A disproportion between  $\max_{\mathbf{x}} |\mathbf{x}|$  and  $\delta$  can obviously lead to an excessive predominance of the threshold and make the scalar product even obsolete. To circumvent the problem of determining the right value for  $\delta$ , we can set it to zero sacrificing one dimension in the hypothesis space (thus  $b=0$ ). Graphically this means that only separating hyperplanes through the origin are considered, reducing the number of potentially solvable problems. In practice, especially in high dimensional spaces as for text documents, this is usually not a very significant restriction, and it additionally renders incremental training possible without parameter tuning.

#### 4.1.4 Dual Form

It is important to see that the weight vector  $\mathbf{w}$  can also be represented as linear combination of the training examples:

$$\mathbf{w} = \sum_{i=1}^m \alpha_i \mathbf{x}_i \quad (4.3)$$

$$h'(\mathbf{x}) = \sum_{i=1}^m \alpha_i \cdot \mathbf{x}_i \mathbf{x} \quad (4.4)$$

assuming  $m$  to be the number of seen training examples and  $\alpha_i \in \{-T, \dots, 0, \dots, T\}$  with  $T$  denoting the number of training epochs. The training examples used in the linear combination, i.e.  $\mathbf{x}_i$  with  $\alpha_i \neq 0$ , are usually called support vectors especially in the context of support vector machines (cf. Section 4.1.6). The perceptron can hence be coded implicitly as a vector of instance or support vector weights  $\alpha = (\alpha_1, \dots, \alpha_m)$  instead of explicitly as a vector of feature weights. This representation is denominated the *dual form* in contrast to the *primal form* given in Eq. 4.1 and is crucial for developing the memory efficient variant in Section 6.2.

#### 4.1.5 Maximum Margin Hyperplane and Problem Hardness

For the perceptron algorithm, the number of errors until convergence depends on the margin between the positive and negative points, i.e. the maximum diameter a separating hyperplane could have (cf. e.g. Bishop 1995, Freund and Schapire 1999, Novikov 1962). The hyperplane that maximizes the margin to the closest positive and negative point is called the *maximum margin hyperplane*, and perceptrons will not necessarily find it. However, the size of the margin is an indicator for the hardness of the learning problem: the smaller the margin the harder it is for the perceptron algorithm to find a good solution.

A different but also important aspect that impacts the difficulty of learning a good hyperplane is the sequence of the training examples. The resulting hyperplane is not invariant against the order of the training examples, which is obvious since the training rule aims at making the model consistent with the current training example. If we order the training examples so that e.g. the positive examples come first, we would expect an increment in the numbers of errors, and hence epochs, until convergence. This disadvantage turns into an advantage in online settings with stream data where the examples have a specific, mostly chronological order and examples become available for training after being tested (cf. Section 4.6.7).

#### 4.1.6 Support Vector Machines

In contrast, Support Vector Machines (SVM) learning algorithms (Cortes and Vapnik 1995) are able to compute the maximum margin hyperplane for given training points. Following Burges (1998)<sup>35</sup> the optimal hyperplane  $\mathbf{w}$  is found by

$$\begin{array}{lll} \text{minimizing} & \frac{1}{2} \mathbf{w}^2 & \\ \text{subject to} & \mathbf{x}_i \mathbf{w} \geq +1 & \forall 1 \leq i \leq m. \mathbf{y}_i = 1 \\ & \mathbf{x}_i \mathbf{w} \leq -1 & \forall 1 \leq i \leq m. \mathbf{y}_i = 0 \end{array} \quad (4.5)$$

<sup>35</sup> We omit the bias  $b$  in the formula since augmenting  $\mathcal{X}$  by a dimension  $a + 1$  which is always  $\delta$  is mathematically equivalent.



which is a convex quadratic programming problem. For the non-linearly separable case, there exists a variant with soft margins  $1 - \xi_i$  or  $-1 + \xi_i$ , respectively, and minimization of  $1/2\mathbf{w}^2 + C \sum_i \xi_i$  with trade-off parameter  $C$ . The problem is often translated into the dual (cf. Section 4.1.4), so that we instead

$$\begin{aligned}
& \text{maximize} && \sum_{1 \leq i \leq m} |\alpha_i| - \frac{1}{2} \sum_{1 \leq i, j \leq m} \alpha_i \alpha_j \mathbf{x}_i \mathbf{x}_j \\
& \text{subject to} && 0 \leq |\alpha_i| \leq C, \forall 1 \leq i \leq m \\
& && \sum_{1 \leq i \leq m} \alpha_i = 0
\end{aligned} \tag{4.6}$$

Roughly speaking, we reduce the problem to finding the support vectors which are placed on the margin ( $\mathbf{x}_i$  with  $\alpha_i \neq 0$ ). This also allows for using the kernel trick, which consists in substituting the dot product  $\mathbf{x}_i \mathbf{x}_j$  by the kernel function  $k(\mathbf{x}_i, \mathbf{x}_j)$ . Computing the non-linear kernel  $k$  corresponds to the dot product  $\Psi(\mathbf{x}_i) \cdot \Psi(\mathbf{x}_j)$  in an augmented feature space. E.g. using the polynomial kernel  $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \mathbf{x}_j)^2$  leads to an expansion from  $\mathcal{X}$  to  $\mathcal{X}^2 = \mathbb{R}^{(a \cdot a)}$ .

In the augmented space it is now much easier to find linear separating hyperplanes, in particular of course for originally non-linearly separable data. In other words, the kernel trick allows for drawing non-linear hyperplanes in the original feature space. Note that this also holds for the perceptron algorithm and generally for all linear classifiers which can be trained and represented in the dual.

#### 4.1.7 Comparison

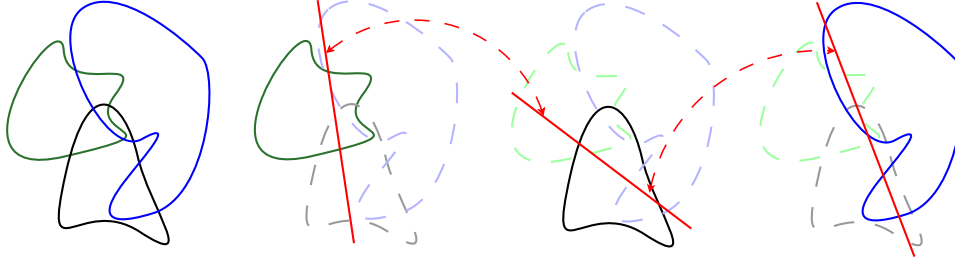
SVMs are generally believed to outperform Perceptrons because they can find the maximal margin hyperplane. However, a closer look reveals that for text classification problems this may not be necessary, and that in turn perceptrons have several advantages.

E.g., as it becomes clear from Eq. 4.5 and 4.6, training SVMs is very sensitive to the size of the training data. Perceptrons in contrast can be trained efficiently in an *incremental* setting, which makes them particularly well-suited for large-scale classification problems such as the *rcv1* benchmark (cf. Section 4.6.1).

The good results of the perceptron variant of Ng et al. (1997) on the smaller Reuters *r21578* confirms the general suitability of perceptrons for text classification. Moreover, Joachims (1998) already found out in an early work that SVMs with linear kernels are sufficient to effectively process texts, which is an additional indication for the actual adequacy for text classification tasks of linear classifiers in general and perceptrons in particular. This hypothesis is backed up by a meta-study in which linear classification algorithms achieve high rankings on *r21578* (Sebastiani 2002, Table 6) and other more recent publications cited in Section 4.5.1.

In addition, important advances were achieved in recent times trying to adapt the perceptron algorithm in order to maximize the margin of the separating hyperplane, without





**Figure 4.1:** Subproblems in *multiclass multilabel perceptrons*: in contrast to binary relevance (cf. Figure 4.1) the separating hyperplanes are interconnected, i.e. they are learned so that the examples in the green (resp. black, blue) cloud have a greater positive margin to the green (resp. black, blue) separating hyperplane than to the other hyperplanes. In other words, that they are more *on their side* than on other sides. The hyperplanes can hence not be trained independently.

losing the advantages of simplicity and efficiency that characterize the perceptron algorithm.<sup>36</sup> A recent overview and comparison of the several different variants concerned with reducing the number of epochs, noisy data robustness, budgeting, etc., is given by Krönke (2011). The basic perceptron variant used commonly throughout in this work can be easily enhanced modularly by any of these extensions with the software package accompanying his work. Alternatively, some researchers have proposed efficient training algorithms for approximating the (linear) maximum margin hyperplane (Fan et al. 2008, Joachims 2006) in linear time, but which are not incrementally trainable such as the perceptrons.

Furthermore, note that contrary to SVMs, and most of the cited perceptron alternatives, the basic perceptron algorithm as presented in this section is parameter-free and hence does not require costly parameter optimizations. This is a well known disadvantage of the otherwise popular SVM training ( $C$  in Eq. 4.6 and kernel parameters, cf. Chang and Lin 2001) .

## 4.2 Binary Relevance Ranking

We train the binary relevance perceptron ensemble exactly as described in Section 3.1 with the only difference that the base learners do not receive  $\mathcal{T}_{rain}^i$  at once but incrementally training instance after training instance. Hence, we obtain  $n$  perceptrons  $\mathbf{w}_i$  which discriminate between their respective classes  $\lambda_i$  and the remaining labels. No perceptron bias is learned, and as outlined in Eq. 4.1, a global threshold of  $\theta = 0$  is used (cf. Eq. 3.4). Ties in the ranking (cf. Section 3.1) are broken randomly to not favor any particular class.

A binary relevance ensemble of perceptrons of this type can be categorized as prototype-based (cf. the discussion about concept learning in Sections 2.5.1 and 3.1),

<sup>36</sup> See e.g. Crammer et al. (2006), Dekel et al. (2005), Freund and Schapire (1999), Khardon and Wachman (2007), Li et al. (2002), Shalev-Shwartz and Singer (2005), Tsampouka and Shawe-Taylor (2007).

---

**Require:** Training example pair  $(\mathbf{x}, P)$ , perceptrons  $\mathbf{w}_1, \dots, \mathbf{w}_n$

```

1: calculate  $\mathbf{xw}_1, \dots, \mathbf{xw}_n$ , loss  $\delta$ 
2: if  $\delta > 0$  then                                ▷ only if ranking is not perfect
3:   calculate error sets  $\mathcal{E}, \mathcal{F}$ 
4:   for each  $\lambda_i \in \mathcal{F}$  do  $\tau_i \leftarrow 0$                 ▷ initialize  $\tau$ 's
5:   for each  $(\lambda_u, \lambda_v) \in \mathcal{E}$  do
6:      $p \leftarrow \text{PENALTY}(\mathbf{xw}_1, \dots, \mathbf{xw}_n)$ 
7:      $\tau_u \leftarrow \tau_u + p$                                 ▷ push up positive classes
8:      $\tau_v \leftarrow \tau_v - p$                                 ▷ push down negative classes
9:      $\sigma \leftarrow \sigma + p$                                 ▷ for normalization
10:   normalize  $\tau$ 's
11:   for each  $\lambda_i \in \mathcal{F}$  do
12:      $\mathbf{w}_i \leftarrow \mathbf{w}_i + \delta \frac{\tau_i}{\sigma} \cdot \mathbf{x}$                 ▷ update perceptrons
13: return  $\mathbf{w}_1 \dots \mathbf{w}_n$                                 ▷ return updated perceptrons

```

---

**Figure 4.2:** Pseudocode of the training method of the MMP algorithm.

since a test point is predicted according to which weight vector  $\mathbf{w}_i$  is closest with respect to the cosine similarity  $\mathbf{xw}_i/|\mathbf{x}||\mathbf{w}_i|$  (Sebastiani 2002).

### 4.3 Multiclass Multilabel Perceptrons

Multiclass Multilabel Perceptrons (MMPs) were proposed as an extension of the binary relevance algorithm with perceptrons as base learners (Crammer and Singer 2003). Just as in binary relevance, one perceptron is trained for each class, and the prediction is calculated via the inner products. The difference lies in the update method: while in the binary relevance method all perceptrons are trained independently to return a value greater or smaller than zero, depending on the relevance of the classes for a certain instance, MMPs are trained to produce a good ranking so that the relevant classes are all ranked above the irrelevant classes. The perceptrons therefore cannot be trained independently, considering that the target value for each perceptron depends strongly on the values returned by the other perceptrons. This difference to BR is emphasized in the illustration in Figure 4.1.

The pseudocode in Figure 4.2 describes the MMP training algorithm. When the MMP algorithm receives a training instance  $\mathbf{x}$ , it calculates the inner products, the ranking and the loss on this ranking in order to determine whether the current model needs an update. For determining the ranking loss, any of the methods of Section 2.7.4 is appropriate, though the output may have to be adapted so that a low value is returned on good rankings. This allows to optimize the ranking in accordance with the used ranking loss. If the ranking is perfect, the algorithm is done, otherwise it calculates the error set of wrongly ordered class pairs  $\mathcal{E}$  (cf. Eq. 2.46). The wrongly ranked classes are also represented in  $\mathcal{F}$  (cf. Eq. 2.50). In the next step, each class that is present in a pair of  $\mathcal{E}$  receives

---

**Require:**

```
    Training example pair  $(\mathbf{x}, P)$ ,
    perceptrons  $\{\mathbf{w}_{u,v} \mid u < v, \lambda_u, \lambda_v \in \mathcal{L}\}$ 
1: for each  $(\lambda_u, \lambda_v) \in P \times N$  do
2:   if  $u < v$  then
3:      $\mathbf{w}_{u,v} \leftarrow \text{TRAINPERCEPTRON}(\mathbf{w}_{u,v}, (\mathbf{x}, 1))$  ▷ train as positive example
4:   else
5:      $\mathbf{w}_{v,u} \leftarrow \text{TRAINPERCEPTRON}(\mathbf{w}_{v,u}, (\mathbf{x}, 0))$  ▷ train as negative example
6: return  $\{\mathbf{w}_{u,v} \mid u < v, \lambda_u, \lambda_v \in \mathcal{L}\}$  ▷ updated perceptrons
```

---

**Figure 4.3:** Pseudocode of the training method of the MLPP algorithm.

a penalty score. This is done according to a selectable penalty function. Crammer and Singer (2003) propose several methods, including a function that returns a value proportional to the difference of the scalar products of both classes. The most successful one, however, seemed to be the uniform update method, where each pair in  $\mathcal{E}$  receives the same score. In the next step, the update weights  $\tau$  are normalized and each perceptron whose class was wrongly ordered is updated.

An example will illustrate the peculiarities of the MMP update method: Suppose that all classes are correctly ordered except for one relevant and three irrelevant classes. The three negative classes are ranked immediately over the positive and under the other positive classes which are at the top. The error set contains three wrongly ordered pairs and according to the uniform update method the positive class will receive in the sum a penalty of 3 and the negatives each 1. Thus the perceptron of the positive class will be updated to a degree three times as great compared to the other three, in accordance with the degree to which it contributed to the wrong ranking. Note that regardless of the used penalty function the positive and the negative classes receive in total the same penalty scores and these are afterwards normalized, so that the degree of the overall model update only depends on  $\delta$ , i.e. on the quality of the ranking. More precisely, the hyperplanes of the perceptrons of the relevant classes are translated by a total amount of  $\delta \cdot \mathbf{x}$ , and the remaining classes by  $-\delta \cdot \mathbf{x}$ . In summary, the degree of the update for a particular perceptron depends 1) on the used penalty method, 2) on how much it contributed to the wrong ranking, and 3) on the general ranking performance.

#### 4.4 Multilabel Pairwise Perceptrons

The decomposition of a multilabel problem using perceptrons as base learner is done exactly as described in Section 3.4 with the difference that Multilabel Pairwise Perceptrons (MLPP) are trained incrementally. This is reflected in the pseudocode in Figure 4.3. Instead of iterating over the pairwise training sets  $\mathcal{T}_{rain}^{u,v}$  and training  $\mathbf{w}_{u,v}$  serially, the procedure in Figure 4.3 obtains one multilabel training instance  $\mathbf{x}$  at each time and passes it the concerned perceptrons with respect to the pairs in  $P \times N$ . TRAINPERCEPTRON cor-

responds to the update in Eq. 4.2. Note that the pairwise models are symmetric, i.e.  $\mathbf{w}_{v,u} = -\mathbf{w}_{u,v}$ .

If we add the virtual label for calibration, we train the additional perceptrons  $\mathbf{w}_{0,1} \dots \mathbf{w}_{0,n}$  as described in Section 4.2 for the binary relevance version of the perceptron ensemble. We denominate this version CMLPP.

The prediction phase was already profoundly described in Section 3.4.3 and 3.4.5. (C)MLPP uses simple voting so that the illustrations in Figure 3.4 and 3.6 apply.

In our particular implementation we distribute  $2 \times 1/2$  votes if a base classifier ties or was not trained (e.g. if two classes are fully correlated). Moreover, we randomly solve ties in rankings, but we use  $v_v \geq v_0$  for deciding if we predict a class  $\lambda_v$  (cf. Eq. 3.25).

## 4.5 Comparison

The next two subsections discuss and compare the differences between the perceptron variants of the BR and LPC decompositions and analyze the computational costs. Most of the advantages and disadvantages already discussed in Section 3.5 also apply for these variants.

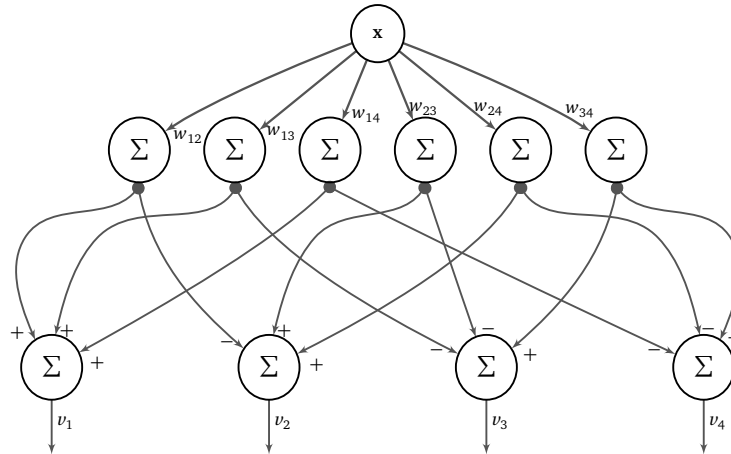
### 4.5.1 Discussion

In particular, we expect a better effectivity of MLPP due to the reduction of the sizes of the subproblems (cf. Section 3.5.1). A simple example illustrates this: imagine two points  $a$  and  $b$  on a line representing the center of the positive and negative points, respectively. We now insert points according to an arbitrary distribution around  $a$  and  $b$ . Let  $\mu(m)$  denote the margin between the negative and positive points depending on the number of inserted points  $m$ . This function is inevitably monotonically decreasing.

Thus it is very likely for a subproblem to have a larger margin than the full BR problem. We have seen in Section 4.1 that the performance strongly depends on the available margin between the points of the binary classes. Indeed, Knerr et al. (1992) observed that the classes of a digit recognition task were pairwise linearly separable, while the corresponding one-against-all task was not solvable with perceptrons. Thus, it can be expected that the MLPP algorithm will also benefit from the pairwise approach.

Since the MMP algorithm is based on the binary relevance binarization, it can also be expected for the pairwise approach to be superior. After all, the MMP algorithm has the same problem space as the binary relevance method: the perceptrons have to find hyperplanes that separates one class from the others, with the difference that the algorithm can translate the hyperplanes along the normal vector and scale the inner product in order to fit correctly in the ranking.

Since the revival of the perceptron algorithm with the publication of the voted perceptrons approach by Freund and Schapire (1999), this learner has repeatedly shown its efficiency and also effectivity in a wide range of domains (cf. Footnote 36). On the popular small Reuters *r21578* text benchmark e.g., an adapted version called CLASSI



**Figure 4.4:** MLPP ensemble represented as artificial neural network. The labels at the arrows indicate the weights of the input connections (multiplication), the full circles denote the sign activation functions. The root node and arrows represent  $a$  input nodes and output connections resp., one for each feature of  $x$ . All other connections are one-dimensional.

outperformed Ripper, one of the most effective rule learners (Ng et al. 1997). An adapted perceptron algorithm referred to as Hieron (Li and Bontcheva 2007) showed to be clearly more efficient but also more accurate than SVMs applied on a hierarchical ontology based information extraction task, which has usually similar characteristics than text classification. However, the non-adapted version of the perceptron with uneven margins (Li et al. 2002) was comparable in absolute numbers but slightly inferior. A similar hierarchical approach was able to outperform binary relevance SVMs on the huge Reuters benchmark *rcv1* (Cesa-Bianchi et al. 2006). Eventually, hierarchical SVMs trained in a very similar way as in Chapter 7 dominated. However, the perceptrons were only trained in one epoch while the SVMs performed the whole optimization process.

Furthermore, one has to always bear in mind that SVMs are not incrementally trainable while perceptrons certainly are. This is a clear advantage especially for large-scale data and for scenarios with real-time learning and classification demands.

It is interesting to note that the MLPP ensemble of perceptrons can be seen as a feed forward neuronal network with one hidden layer and fixed connections between the nodes. The pairwise perceptrons correspond to nodes in the middle layer with the indicator threshold function as activation function and the voting mechanism corresponds to the output nodes at the bottom layer. An illustration is given in Figure 4.4. In the same manner BR and MMP can be represented as fixed artificial neuronal networks without hidden layer.

#### 4.5.2 Computational Complexity

The complexities of BR and LPC were already analyzed and compared in detail in Sections 3.1.1 and 3.4.6. However, an analysis of BR, MMP and MLPP with perceptrons is

**Table 4.1:** Computational complexity of perceptron ensembles given in expected number of dot products and additions of vectors per instance, and number of vectors for memory, respectively. Given in terms of number of total labels  $n$ , current  $|P|$  and average labelset size  $d$  and losses from Section 2.7.

	training time	prediction time	memory requirement
perceptron	$1 + \text{ERR} = \mathcal{O}(1)$	1	1
BR	$n(1 + \text{HAMLoss}) = \mathcal{O}(n)$	$n$	$n$
MMP	$n + \text{MARGIN} + \text{ISERR} = \mathcal{O}(n)$	$n$	$n$
MLPP	$ P (n -  P )(1 + \text{ERR}) = \mathcal{O}(dn^2)$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
$\frac{\text{MLPP}}{\text{MMP}}$	$ P  = \mathcal{O}(d)$	$\frac{n-1}{2}$	$\frac{n-1}{2}$

particularly interesting for three reasons: Firstly, the costs were given very abstractly in numbers of training examples or using very abstractly estimated complexities. The analytically simple perceptron algorithm, which is common to all considered approaches, allows for a very concrete analysis. Secondly, the presented approaches are incrementally trainable and hence the following analysis can be considered as an extension. Moreover, perceptrons learn in linear time and predict in constant time, so this analysis shows explicitly the case where  $p = 1$  and  $q = 0$ . And thirdly, although MMP is based on BR it has a different training with possibly different runtime constraints which deserves a consideration itself.

We use the same notation as in previous analyses. In addition,  $a$  denotes the number of attributes and  $a'$  the average number of features not zero (size of the sparse representation of an instance).

Except for the last part, we will indicate the runtime dependencies in terms of perceptron prediction and update operations, since a scalar product  $\mathbf{w}\mathbf{x}$  requires nearly the same amount of arithmetic additions and multiplications as an update  $\mathbf{w} + \tau\mathbf{x}$ . However, if the factor  $\tau$  is zero, there may be indeed a deviation. We ignore operations that have to be performed by all algorithms such as sorting or internal real value operations. Additionally, we will present the complexities per instance since all algorithms are incrementally trainable.

We explicitly only consider the common variant of the perceptrons with simple weight vectors  $\mathbf{w}$ . Please refer to Chapter 6 and particularly Section 6.3 for the dual variant.

#### 4.5.2.1 Memory Requirements

BR and MMP follow prototype-based approach, so they have to keep one perceptrons for each class in memory, leading to  $n \cdot a = \mathcal{O}(na)$  memory space. In contrast, the pairwise approaches require one perceptron for each of the  $\frac{n(n-1)}{2}$  pairs of classes, hence we need

$\mathcal{O}(n^2a)$  memory. In addition, the calibrated versions require an overhead of  $n$  perceptrons for the comparisons with the artificial label.

Since all perceptron ensembles are online-learners, we do not need to store the whole training set in memory, so the requirements are reduced from  $ma$  to the  $a$  needed for the current training instance.

#### 4.5.2.2 Training

For processing one training example  $n$  dot products have to be computed by BR, plus at most the same amount of vector additions if there was a prediction error. Following Eq. 2.43, the costs are  $n(1 + \text{HAMLoss})$ .<sup>37</sup> MMP has to update each of the wrong prototypes  $\mathbf{w}_i, \lambda_i \in \mathcal{F}$  in addition to the initial prediction. Conveniently,  $|\mathcal{F}|$  amounts to  $\text{IsErr} + \text{Margin}$  (cf. Eq. 2.50), so that exactly  $n + \text{IsErr} + \text{Margin}$  operations are required.

The MLPPs require  $|P|(n - |P|)$  dot products, one for each associated perceptron. Assuming an average prediction error of the base perceptrons of  $\text{Err}$ , the costs amount to  $|P|(n - |P|)(1 + \text{Err})$ . Unfortunately, it is not possible to obtain a direct relation between  $\text{Err}$  and a multilabel metric in Section 2.7 such as for BR and MMP. The investigation of this relationship remains for the future.

Assuming similar loss rates for all algorithms, i.e.  $(\text{IsErr} + \text{Margin})/n \approx \text{HAMLoss} \approx \text{Err} \approx \delta$ , MLPP requires  $|P| - |P|/n \leq |P|$  times the number of operations of MMP or BR, hence on average  $\leq d$ , confirming the analysis in Section 3.4.6. Assuming perfect prediction  $\delta = 0$  and the worst case  $\delta = 1$ , respectively, for the pairwise and both prototype-based approaches, and assuming<sup>38</sup>  $|P| \leq n/2$ , the costs ratio  $r$  between MLPP and BR/MMP is bounded by

$$\frac{1}{4}|P| = |P|\frac{1/2n}{2n} \leq |P|\frac{n-|P|}{2n} \leq r \leq |P|\frac{2(n-|P|)}{n} < |P|\frac{2n}{n} < 2|P| \quad (4.7)$$

If the calibrated version CMLPP is used, we have to add the BR operations. For the average number of operations per instance on the whole training set, we can substitute  $|P|$  by  $d$  in the statements.

Thus, assuming similar loss rates, the pairwise training will be only  $d$  times slower on average than the BR algorithm (or  $d + 1$  respectively for the calibrated version) despite training a quadratic number of base classifier.

#### 4.5.2.3 Prediction

During prediction the one-per-class approaches achieve  $n$  computations for one instance, since both use the same model space. For the pairwise approach all  $n(n-1)/2$  perceptrons

<sup>37</sup> We simplify the notation and write  $\delta$  instead of  $\delta(P, \mathbf{r})$  for losses computed on the prediction for a training instance  $\mathbf{x}$ .

<sup>38</sup> This is no restriction, since otherwise we could just use  $|N|(n - |N|)$  for the particular estimation, which corresponds to inverting the problem. The maximum number of operations for MLPP is reached with  $|P| = n/2$ .



have to be evaluated, leading to  $\mathcal{O}(n^2)$  computations. The ratio between both decomposition philosophies is hence  $(n - 1)/2$ . If calibration is used,  $n(n + 1)/2$  perceptrons and thus  $(n + 1)/2$  times more operations have to be performed.

#### 4.5.2.4 Sparsity of Feature Vectors

If the feature vectors of the training and test instances are sparse, i.e. the average number of components  $x_i \neq 0$  over all  $\mathbf{x}$ , which we represent with  $a'$ , is low, perceptron based approaches and generally linear classifiers can benefit computationally from an effective representation of the instances. A sparse data structure only stores the non-zero components and information about the indices gaps. This can be implemented e.g. by two vectors  $\in \mathbb{R}^{a'}$  or a linked list.

While a sparse representation can save memory space when the whole training set has to be stored, no reduction can be achieved for the perceptron ensembles since a high density for these vectors is very likely. Hence,  $na$  numbers have to be maintained in memory by BR and MMP, and  $\mathcal{O}(n^2a)$  for MLPP and CMLPP.

For training and prediction in contrast, we obtain the number of arithmetic float operations by multiplying the stated costs in number of perceptron operations by  $a'$ .

### 4.6 Evaluation

The *rcv1* collection of Reuters news articles with 103 classes and 804,414 examples is the largest and, in our opinion, most interesting test bed for the pairwise approach presented in this chapter because the high number of documents particularly puts the claimed scalability to test. We will present a detailed analysis of the results on this dataset, and, in Section 4.6.8, show a brief summary of results on the other datasets, which will essentially confirm the results on the *rcv1* benchmark. Statistics on *rcv1* can be found in Section 6.1, which particularly focuses on the distribution of the labels and compares *rcv1* to the *EUR-Lex* dataset (which is more suitable for testing the scalability to a large number of labels) and in the datasets overview in Section 2.9.1.

#### 4.6.1 Experimental Setup

As already outlined in Section 2.9.1, we split the *rcv1* dataset into 535,987 training and 268,427 testing documents. Several tests with different values for the number of attributes and different methods for term weighting and feature selection were done in a systematic way in order to determine the most appropriate settings for both algorithms on the Reuters *rcv1*. Typically, we used MMPs to reduce the number of candidates and, among the remaining candidates, we picked a setting that worked well for both. The following settings proved to generally provide good results and to allow a fair and representative comparison: we used the common TF-IDF term weighting method (Salton and Buckley 1988, Sebastiani 2002) and used the first 25,000 features ordered by their



**Table 4.2:** Comparison on the *rcv1* test set. The first block reports ranking metrics, while the second and third one shows pseudo and real bipartitioning performance. All measures except `ERRSetSize` and `MARGIN` are given in percentages, `HAMLoss` is presented inverted. Recall, precision, and F1 were (label and example based) micro-averaged. In the first block the less the better except for `AVGP`.

	NB	BR	MMP	MLPP	CMLPP
<code>IsERR</code>	51.79	35.87	29.35	28.14	<b>27.36</b>
<code>ERRSetSize</code>	6.285	7.614	2.801	1.920	<b>1.904</b>
<code>RANKLoss</code>	1.709	2.529	0.687	0.478	<b>0.472</b>
<code>MARGIN</code>	4.513	5.833	2.120	1.453	<b>1.438</b>
<code>ONEERR</code>	12.85	4.022	3.750	2.964	<b>2.902</b>
<code>AVGP</code>	82.26	90.00	92.82	93.67	<b>93.81</b>
<code>F1<sub> P </sub></code>	73.88	81.40	86.74	87.74	<b>87.99</b>
<code>PREC<sub>d</sub></code>	71.61	78.86	81.92	82.56	<b>82.74</b>
<code>REC<sub>d</sub></code>	66.51	73.24	76.08	76.67	<b>76.85</b>
<code>F1<sub>d</sub></code>	68.96	75.95	78.89	79.51	<b>79.68</b>
<code>1 - HAMLoss</code>	–	98.74	–	–	<b>98.97</b>
<code>PREC</code>	–	80.15	–	–	<b>86.77</b>
<code>REC</code>	–	<b>79.70</b>	–	–	79.33
<code>F1</code>	–	79.93	–	–	<b>82.88</b>

document frequency. All parameters of the preprocessing methods were only computed on the training set to ensure that no information from the test set enters the training phase. This setting was also retained in subsequent experiments with *rcv1* and, with the exception of differing number of features, also on other text collections for which we did the preprocessing.

For the MMP algorithm we used the `IsERR` loss function and the uniform penalty function. This setting showed the best results in the work of [Crammer and Singer \(2003\)](#) on the *rcv1* dataset and our experiments confirm this. All perceptrons were initialized with random values.

We performed also tests with the binary relevance method and a multilabel variant of the multinomial Naive Bayes (NB) algorithm (cf. [Mitchell 1997](#), Sec. 6.9) in order to provide a baseline. In one of our first experiments we counted the TF-IDF instead of the term frequency values for the Naive Bayes. We found out that by using this additional information about the overall relevance of each term the accuracy even doubled for some losses. We report therefore these improved results.<sup>39</sup>

<sup>39</sup> Note also that using the Naive Bayes as base classifier for a pairwise binarization is pointless as it results in the normal Naive Bayes ([Sulzmann et al. 2007](#)).

---

### 4.6.2 Ranking Quality

The ranking results are presented in the first block of Table 4.2 for a series of ranking losses (cf. Section 2.7.4).

The binary relevance classifier is clearly outperformed by all other approaches, even by the baseline Naive Bayes classifier for some metrics. This is not surprising, as it is the only method that is not concerned with optimizing a ranking. However, the pairwise algorithms MLPP and CMLPP also clearly outperform the MMP algorithm, which is essentially a BR variant with a training procedure that aims at optimizing their ranking performance (all differences are highly statistically significant with at least  $\alpha = 0.1\%$  according to the Wilcoxon signed-rank test, see also further below).

Especially on the losses that directly evaluate the ranking performance the improvement is quite pronounced. On average, (C)MLPPs increase the number of correctly ranked relevant and irrelevant class pairs by almost one pair per example (ERRSETSIZE). Similarly, the margin between the positive and negative classes is improved by more than half a class (MARGIN). MLPP also has a  $\approx 0.8\%$  ( $\approx 2,100$  documents) advantage in the percentage of examples for which the top rank is correct (ONEERR). For the ISERR, the advantage is less pronounced. Typically, a perfect classification is more likely to occur on documents that have a small number of labels, whereas on documents with an increasing number of labels the ISERR performance decreases rapidly. Thus, ISERR focuses more on the performance on cases where there is not much to rank. The AvgP measure yields a similar gain.

It is particularly important to note that MLPPs outperform MMPs in terms of ISERR, although MMPs were trained to directly optimize this loss function, whereas MLPPs are independent of a particular loss function. This holds also if MMPs are trained to optimize a different loss. For example, the best MMPs trained to optimize MARGIN yield an average MARGIN-loss of 1.95.

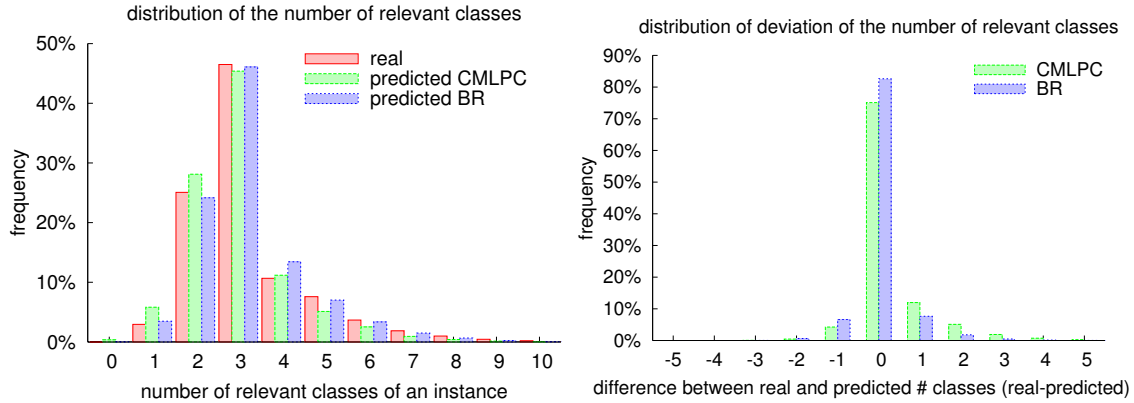
Surprisingly, CMLPP also improves the ranking performance over MLPP. Due to the large number of test examples, these seemingly small improvements are statistically highly significant.<sup>40</sup> Apparently, the information provided by the introduction of the artificial calibration label not only allows to split the classes into relevant and irrelevant, but the additional  $n$  binary models that are learned by CMLPP also help to somewhat improve the ranking of the other classes. However, large improvements cannot be expected, because the additional preferences involving the calibration label can increase the voting count of each label by at most 1, which allows only minor changes in the ranking positions.

### 4.6.3 Calibration Performance

The (label and example-based) micro-averaged evaluation measures in the last block of Table 4.2 can only be computed for the binary relevance classifiers and the calibrated

---

<sup>40</sup>  $p \ll 0.0001$  according to the ANU sign test (cf. Section 2.10.1) and considering each test instance as an independent sample.



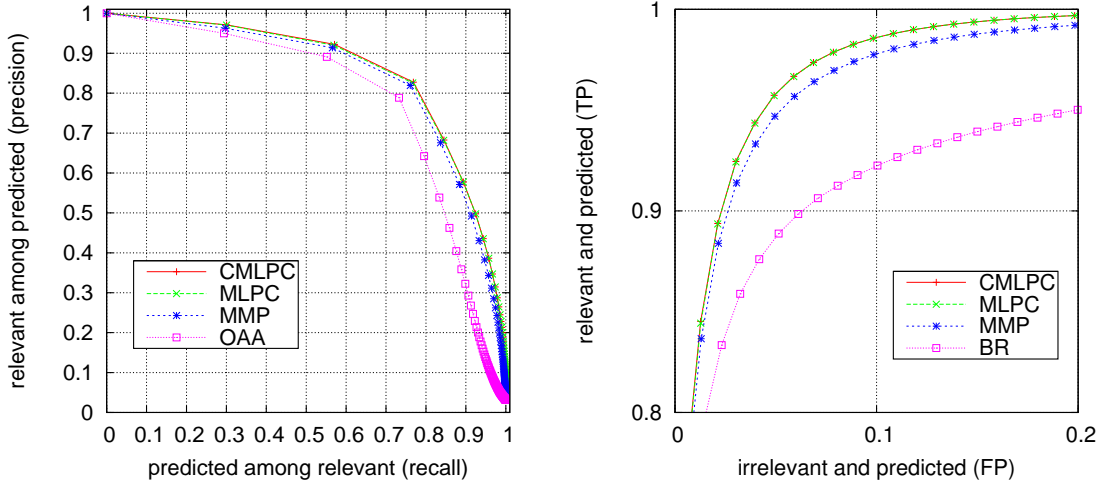
**Figure 4.5:** Predicted and actual number of classes for the calibrated ranking and for the binary relevance approach on the *rcv1* dataset.

pairwise multilabel perceptrons, because the other approaches are restricted to rank the labels and do not separate them into relevant and irrelevant labels. The results show that CMLPP competently splits between relevant and irrelevant classes: for every prediction there is on average approximately only one class that is placed on the wrong side of the boundary (either via bad ranking or via a bad setting of boundary). For BR, the performance is similar but somewhat worse.

Figure 4.5 shows the actual distribution of the number of labels for each example, and the distributions that result from the predictions of BR and CMLPP (irrespective of whether the predictions are correct or not). Obviously, both algorithms follow the original distribution quite closely. CMLPP predicts the correct number of classes in about 75% of the cases, in more than 90% the deviation was one class or less. Interestingly, these numbers are even higher for the binary relevance ranking, even though its overall performance is worse because of its bad ranking performance as discussed above. Thus, the calibration point of the CMLPP rankings is closer tied to the ranking performance of the algorithm, while for BR the two appear to be more independent. In general, a small bias towards underestimating the number of labels is noticeable for CMLPP. This is also shown by CMLPP’s recall, the only measure for which the pairwise approach is slightly beaten (by BR).

In order to get an idea of the quality of the predicted boundary between relevant and irrelevant examples, the second block in Table 4.2 shows a comparison of the boundaries predicted by BR and CMLPP to a fixed boundary of three (the median value), and the real boundary (cf. Section 2.7.4). This also allows to compare against the ranking algorithms, so to say as if they were able to produce a bipartitioning.

The results show that CMLPP clearly improves over the median ( $\text{PREC}_d$ ,  $\text{REC}_d$  and  $\text{F1}_d$ ), but it also does not come near the performance using the real boundary (idealistic  $\text{F1}_{|P|}$ ). Remind that for this boundary recall, precision and F1 are equal and that  $\text{F1}_{|P|}$  is not necessarily the maximal achievable value (cf. Section 2.7.4). The comparison to MLPP and



**Figure 4.6:** Micro averaged recall/precision and ROC curves on the *rcv1* dataset.

MMP confirms once more that CMLPP produces the best rankings, with a clear advantage over MMP and BR. As already indicated, the binary relevance ranking BR has, for the predicted boundary, a small advantage in terms of recall due to the somewhat more cautious prediction of the boundary by CMLPP, but clearly suffers from a lack in precision and a bad ranking performance, as can again be seen from the results for the median and real boundaries.

In order to visualize the performance for different boundaries and the typical recall/precision trade-off, we plotted these two measures and the true and false positive rates for all possible boundaries  $b = 0 \dots 103$  in Figure 4.6 (cf. Section 2.7.4).

The resulting recall/precision plot in the left part shows that the curve of BR is completely dominated by the MMP curve, which in turn is dominated by the MLPP and the CMLPP curves.

The complete ROC-curve exhibits a qualitatively similar behavior. In the right part of Figure 4.6 we have enlarged the upper left area  $[0.0 - 0.2, 0.8 - 1.0]$  to make these small differences more visible. There are also regions in the graph where very small differences between CMLPP and MLPP are noticeable in the graphs, but the resolution of the graphs is too small to show them.

#### 4.6.4 Computational Costs

We measured the run-time in order to compare it to our analysis of the computational complexity in Section 4.5.2. We found it most convenient to measure an amount of processed operations instead of an amount of (CPU-)time, since in this way it is guaranteed to be independent from external factors such as logging activities and others not part of the basic algorithm, suboptimal routines in the underlying workbench, activities of

**Table 4.3:** Comparison of the number of operations of the different algorithms for training and testing, in number of arithmetic and vector operations, and the memory requirements in kilobytes.

	BR	MMP	MLPP	CMLPP
train arithm. op.	4,237,467,068	4,307,889,941	13,287,753,360	17,525,186,974
test arithm. op.	2,080,697,850	2,080,697,850	106,115,590,350	108,196,288,200
train vector op.	55,896,832	56,680,708	174,184,241	229,390,902
test vector op.	27,647,981	27,647,981	1,410,047,031	1,437,695,012
memory in kb	362,534	366,392	862,414	872,900

the operation system etc. Since all the algorithms used are based on the perceptron algorithm, a basic operation that is appropriate to compare the results of the different algorithms can easily be found. Following our analysis, we report vector operations (dot products and additions) as well as the caused arithmetic floating number operations. Other operations such as comparisons and sortings were ignored.

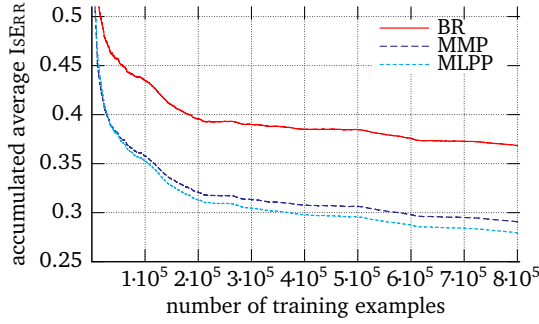
Table 4.3 shows the number of operations in the training and testing process used by the respective algorithms. The ratio of training operations between the binary relevance and the pairwise comparison approach averages 3.08 and therefore corresponds to the average number of relevant classes per example in the used dataset. Analogously, the ratio between testing with MLPP and MMP is exactly  $103^{-1/2} = 51$ . Also note that the complexity of the CMLPP approach equals the sum of the complexities of the MLPP and the BR approaches.<sup>41</sup> All these observation are in agreement with our analysis in Section 3.4.6 and 4.5.2.

In contrast, the true memory requirements is not reflected in the analysis due to the overhead of the runtime environment. With 25,000 features and 4 bytes for each real number, the relation should theoretically be 9.8 to 501 megabytes. The throughput of training document per second was 1175/1252 for BR/MMP, and 605/772 for C/MLPP in our experiments on a 2 GHz Dual Core Opteron. BR/MMP processed 758/849 test documents per second while C/MLPP achieved respectable 72/74.

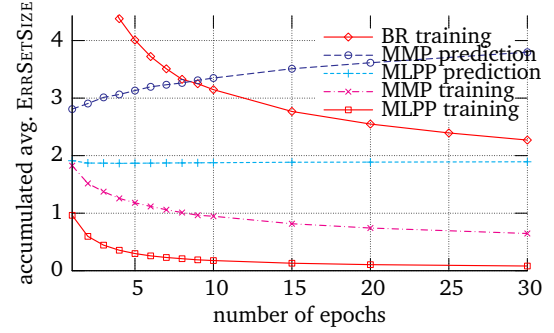
#### 4.6.5 Learning Curve

A learning curve shows how quickly a learner is able to adapt its model to the data presented. For incremental learners, it is often used to show the learning progress. Before a new example is added to the training set, it is first tested. The learning curve then shows the accumulated loss over the processed training instances, divided by the number of training examples. The result for the IsErr loss on the *rcv1* data can be seen in Figure 4.7. Only MLPP is shown for convenience. It is shown that with an increasing number of examples, MLPPs accumulate a clear advantage. However, in the beginning, the differences are less pronounced and the MMP algorithm also seems to have a somewhat

<sup>41</sup> For the training complexities this match is only approximate due to different initialization vectors.



**Figure 4.7:** Learning curve for the full dataset (IsERR).



**Figure 4.8:** Error on training and test data depending on the number of epochs.

better performance in this region. For the ranking loss `RANKLoss` (not shown here) the graph look very similar, except that here MLPPs clearly have a better performance from the start. This result and even the comparison in terms of `IsERR` is remarkable because the perceptrons of the MLPP are trained on fewer examples, which may be particularly problematic in the beginning of the training phase.

#### 4.6.6 Overfitting Analysis

In order to evaluate the overfitting property, both algorithms were trained in several epochs over the training set. [Crammer and Singer \(2003\)](#) observed that the performance of the MMP algorithm became worse with an increasing number of epochs. Our results confirm this observation: While the evaluation on the training data indicates a better adaptation, the performance on the test data decreases (Figure 4.8). For the MLPP algorithm the better adaptation to the training data is also clearly observable, it quickly reaches losses near 0, but in contrast to MMP, the results on the test data remain stable. We interpret this as evidence that pairwise decomposition of the problem does in fact fit the problem structure, i.e., that the classes here are in fact pairwise linearly separable. MLPPs learn these linear decision boundaries after the first epoch through the training examples so that further training is not necessary (but can also not lead to more overfitting).

#### 4.6.7 Concept Drift Analysis

Our presupposition is usually that the classification mapping is static and does not change over time (cf. Section 2.2). But in practice this assumption does often not hold. In particular in text classification, one often can observe that the notion or semantic of a particular class changes over the time. Sometimes it may also happen that several categories are merged into one, or conversely, one category is divided into several subcategories. This phenomenon is usually called *concept drift* and does typically appear in applications of *stream data classification*.

The *rcv1* is very appropriate in order to analyze the robustness of a learning algorithm since it contains more than 800,000 documents which are chronologically ordered. But since no concrete concept drift is known for this dataset, we artificially introduce it. Following [Kong and Yu \(2011b\)](#), we shift the mapping of the instances  $\mathcal{T} = \langle (\mathbf{x}_i, \mathbf{y}_i) \mid 1 \leq i \leq m \rangle$  as follows

$$\mathbf{y}'_i = \begin{cases} \mathbf{y}_i & , \text{ with probability } p = \frac{m-i}{m} \\ (y_{i,m}, y_{i,1}, \dots, y_{i,m-1}) & , \text{ otherwise} \end{cases} \quad (4.8)$$

resulting in the new drifted example set  $\mathcal{T}' = \langle (\mathbf{x}_i, \mathbf{y}'_i) \mid 1 \leq i \leq m \rangle$ .<sup>42</sup>

[Kong and Yu \(2011b\)](#) very recently proposed to use ensembles of random decision trees for learning stream data with concept drifts. The main idea is to generate  $k_1$  RDTs with random attribute tests at the inner nodes and maximal depth  $k_2$ . Comparably small values of  $k_1$  and  $k_2$ , around 10 or 20 and maximally 100, are sufficient in practice. During the extremely fast training, the leafs incrementally collect statistics about the label distributions  $\mathbf{y}$  and the labelset sizes  $|\mathbf{y}|$  of the instances which passed all tests to the leafs. Hence, each RDT predicts an average distribution and cardinality, which is subsequently averaged over all trees.

RDTs are very suitable for data with a high number of examples and labels, since the costs are bounded by the selection of  $k_1$  and  $k_2$ . However they are not appropriate for data with a high number of possibly sparse features, as we will also see in the following experiments. This is because a RDT tree can maximally cover  $k_1 \cdot k_2$  feature dimensions with their tests, and increasing  $k_1$  and particularly  $k_2$  quickly leads to a extreme deceleration. But for small datasets like *yeast* and *scene*, they obtain very good results, often outperforming SVMs, but using only 10% to 1% of training and testing time.<sup>43</sup> An additional decision tree learner for stream data was proposed by [Read et al. \(2010\)](#), but unfortunately the used Hoeffding trees are not suitable for concept drifts. Recently, [Spyromitros Xioufis et al. \(2011\)](#) proposed to use a windowing mechanism over the positive and negative examples of the base learners of a BR ensemble, which is particularly suitable for nearest neighbor learners.

For the particular case of concept drifts in stream data, [Kong and Yu](#) implemented a technique that subsequently decreases the weight of previous examples. More specifically, the weight of an example decreases by half after a predetermined number  $h$  of subsequent training examples. The integration of the half-life parameter is straightforward for perceptrons by changing the update of the weight vector (Eq. 4.2) to  $\mathbf{w}_{i+1} = 2^{-\frac{1}{h}} \mathbf{w}_i + \alpha_i \mathbf{x}_i$  or

$$\mathbf{w}_{i+1} = 2^{-\frac{i-j}{h}} \mathbf{w}_i + \alpha_i \mathbf{x}_i \quad (4.9)$$

<sup>42</sup> The ordering of the classes is not randomized in our version of *rcv1*, so it is possible that a certain bias was introduced towards drifting to hierarchical close labels or labels similar in size.

<sup>43</sup> [Zhang et al. \(2010b\)](#) and own preliminary experiments with *yeast*, *scene*, *emotions* (cf. Section 2.9.1) not shown here.



respectively if  $j$  was the last index with  $\alpha_i \neq 0$ . After  $m$  training examples, this results in  $\mathbf{w}_{m+1} = \sum_{i=1}^m 2^{-\frac{m-i}{h}} \alpha_i \mathbf{x}_i \mathbf{x}$ . It is easy to see that this corresponds to an increasing learning rate of  $\eta_i = 2^{\frac{i}{h}}$  (cf. Section 4.1.2).

Kong and Yu (2011b) reported an important improvement by assigning a half-life of 200 to the training examples. Particularly for *rcv1*, the improvement was from approx. 0.4 to 0.1-0.14 in terms of RANKLOSS. However, in our experiments using this parametrization of the examples substantially harmed the performance of CMLPP as well as RDTs, as can be seen in Figure 4.9. We adapted the RDT library of Zhang et al. (2010b) in order to support the half-life parameter and tried out different combinations for the number of features, the maximal depth of the trees and the size of the ensemble, following also the recommendations of both publications. The best combination for RDT on *rcv1* was to use 2500 features and 20 trees with a depth of 100. Similarly to the setting of Kong and Yu, RDTs and CMLPP were trained on the first  $67034 \cdot j$ ,  $1 \leq j \leq 11$  examples and tested on the following 67034 ones so that we obtained eleven points for each learning algorithm.

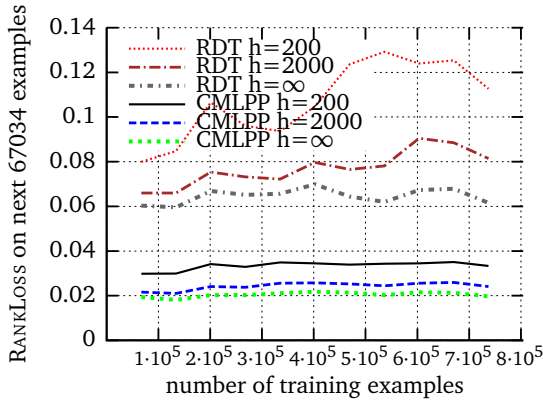
We can see that changing the weights of the training examples clearly harms the performance for both algorithms. The faster the decrease, the more pronounced is the increase in ranking loss. Figure 4.10 shows only the CMLPP variants similarly to Figure 4.7 with the average accumulated RANKLOSS. The last curve additionally shows the average of the previous 10000 ranking losses (obtained in the same way, by testing before training) for an infinite half-life, i.e. the default setting. It can be again clearly observed that a constant learning rate is the best option in this particular setting. It is of particular interest that the RANKLOSS-curves of CMLPP are almost linear and present only a slight ascending slope though the mappings are completely shuffled in the end. This demonstrated the robustness of CMLPP in this particular setting.

Regarding the contrary behavior of RDTs, and also MLPPs, than reported by Zhang et al. (2010b), we cannot exclude an error in the implementation, but we consider it unlikely since the same effect appears for both algorithms. On the other hand, the observations in our experimentation seem reasonable since, even though the drift is very radical, it evolves very slowly during more than 800000 examples. Roughly speaking, after a proposed half-life of 200, the *example base* expectedly becomes only to  $200/804413 \approx 2.5\%$  *more often wrong* than 200 examples before, but the examples have already lost half of its weight. Intuitively, the drifting rate and the decrease rate of the examples weights should correspond. As the curves in Figure 4.10 show, the half-life should lie even over 20000 for the proposed drifting setting. Nevertheless, this aspect should be investigated in further, more varying settings. Moreover, we simply adopted the simple technique also employed by Zhang et al. (2010b), but perceptrons were already investigated under the focus of concept drifts, e.g. by Dekel et al. (2005) and their Forgetrons.

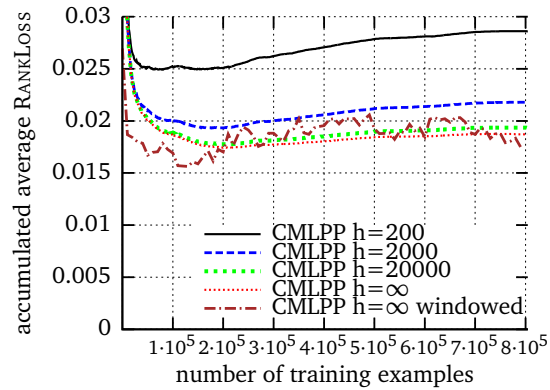
#### 4.6.8 Results on other Datasets

In order to evaluate the algorithms on other datasets, we performed a few quick experiments on the older and smaller Reuters *r21578* dataset (11367 examples, 120 classes,





**Figure 4.9:** Comparison of random decision trees and CMLPP on drifted *rcv1* with varying half-life parameter.



**Figure 4.10:** Learning curve of CMLPP on drifted *rcv1* with varying half-life parameter.

on average 1.26 labels per example, with similar preprocessing as *rcv1* and 10,000 features), and, to represent other application domains, the popular *yeast* (2417 examples, 103 features, 14 classes, on average 4.24 labels per example) and *scene* (2000 examples, 294 features, 5 classes, on average 1.24 labels per example) datasets (cf. Section 2.9.1).

Previous experiments showed that both non-text problems seem to be hardly linearly separable even using pairwise decomposition and that both algorithms appear to not apply well to this sort of problems (Loza Mencía and Fürnkranz 2007b). As the results were inconclusive, we simulated the use of a polynomial kernel of second degree by adding all possible pairwise products of the original features to the feature vector. This allows us to translate the problem into a higher dimensional space where it is more likely linearly separable without having to modify our implementation and analysis. Note however that it is generally possible to use any kernel function in combination with perceptrons (cf. Section 4.1.6).

The results for 10 epochs (i.e. 10 iterations over the training data) in the case of *r21578* and 100 epochs for *yeast* and *scene* over the training data and ten-fold cross-validation are shown in Table 4.4.<sup>44</sup>

For the sake of compactness and due to correlation of the results in practice, only four representative measures are presented for the additional datasets. The results for the Naïve Bayes baseline were also omitted.

Essentially, the results confirm the key findings on the large *rcv1*, namely that

<sup>44</sup> The results may slightly differ from those of Fürnkranz et al. (2008) and Loza Mencía and Fürnkranz (2008c) since they were recently redone in order to perform supplementary statistical tests. We applied the Wilcoxon signed-rank test (cf. Section 2.10.2) on the averages of the cross validation. Some of the measures reported in Table 4.2 were omitted since they did not provide any additional insights. For the passed Friedman test on the averages on the cross validation folds results (one observation per dataset instead of 10), we can conclude that  $BR < MLPP$  (10%) / CMLPP (5%) for RANKLoss and AvgP with the Bonferroni-Dunn test despite the small number of samples (3) and classifiers (4).

**Table 4.4:** Comparison on the *r21578*, *scene* and *yeast* test sets, similarly to Table 4.2. The best result in each row is indicated in bold font, significant differences to CMLPP are indicated with  $-/+$ (10%),  $--/++$ (5%),  $_{-}/_{+}$ (1%).

(a) <i>r21578</i>				
	BR	MMP	MLPP	CMLPP
RANKLoss	2.977 $_{-}$	0.453 $_{-}$	0.250 $_{-}$	<b>0.239</b>
AvGP	91.59 $_{-}$	95.46 $_{-}$	95.47 $_{-}$	<b>95.89</b>
PREC	78.38 $_{-}$	–	–	<b>87.98</b>
REC	<b>85.59</b> $_{+}^{++}$	–	–	83.79

(b) <i>scene</i>				
	BR	MMP	MLPP	CMLPP
RANKLoss	8.165 $_{-}$	7.822 $_{-}$	7.435	<b>7.285</b>
AvGP	85.64 $_{-}$	85.82 $_{-}$	86.45 $_{-}$	<b>86.79</b>
PREC	71.80	–	–	<b>71.83</b>
REC	71.21 $_{-}$	–	–	<b>74.20</b>

(c) <i>yeast</i>				
	BR	MMP	MLPP	CMLPP
RANKLoss	22.73 $_{-}$	21.03 $_{-}$	<b>17.49</b>	17.54
AvGP	70.41 $_{-}$	71.39 $_{-}$	<b>75.09</b> $_{+}$	74.98
PREC	60.47 $_{-}$	–	–	<b>62.37</b>
REC	59.07 $_{-}$	–	–	<b>63.31</b>

1. the pairwise ranking methods outperform the one-against-all ranking methods. Especially on *r21578* the pairwise approaches clearly outperform the BR based classifiers. On *scene* and *yeast*, the margin decreases, but (C)MLPP is still clearly superior and all differences (to the calibrated version) are statistically significant. Nevertheless the problem seems to be generally hard to handle even with the usage of kernels since the losses are quite high. It is also interesting to note that the distance between BR and MMP clearly decreases for the two hard datasets with small number of classes.
2. adding the calibration label to the ranking does not systematically deteriorate performance (in the *yeast* dataset it lead to worse results, on the *scene* and *r21578* it helped)
3. the calibrated pairwise method outperforms the binary relevance predictor in terms of bipartition metrics. Although BR can beat CMLPP on *r21578*'s recall, which is due to the next point, for the remaining datasets and for remaining bipartition metrics F1, HAMLoss and Acc (not shown here), BR is dominated by the pairwise approach.

- 
4. *r21578* confirms that the calibrated pairwise method is a bit more conservative in predicting the number of relevant labels (lower recall than BR and on average 0.17 less predicted labels), but *scene* and *yeast* shows the opposite picture (on average 0.04 and 0.16 more predicted labels). This matter will be discussed more extensively in Section 4.6.9.

The calibrated pairwise method combines the advantages of both methods: it utilizes the improved ranking capabilities of the pairwise approach, but, through its calibration label, is able to make a good prediction where in the predicted ranking the labels should be separated into relevant and irrelevant labels.

Although we see the main application field of the MLPP algorithm in the efficient solving of large scale (in number of examples as well as features) problems such as text classification, where feature vectors are additionally very sparse, the reported results are competitive with other published results on the *yeast* and *scene* dataset (cf. e.g. Veloso et al. 2007, Zhang and Zhou 2006).

An extended comparison between CMLPP and BR on a broader range of datasets in Section 5.4.2 again confirm the advantages of the pairwise approach. However, in contrast to the comparison presented here, the employed variant of CMLPP does only allow an evaluation of the bipartitioning performance.

#### 4.6.9 Discussion on Cardinality Prediction

On first sight, it seems that the underestimation of the cardinality depends on the number of labels, since the Reuters datasets both are large scale in this respect. However, another explanation could be a dependency on the label density  $d/n$ , which is very low for these two benchmarks: A true class  $\lambda_u$  has to collect at least  $v_0 \approx |N|$  votes from the  $|N|$  *decisive* classifiers  $h_{u,v}$ ,  $\lambda_v \in N$  and from the  $|P|$  *indecisive* pairings with  $\lambda_v \in (P \setminus \{\lambda_u\}) \cup \{\lambda_0\}$  in order to be predicted as positive. If the density is high(er), the sum of the indecisive classifiers is also statistically high(er) (expectedly  $|P|/2$ ). Consequently,  $\lambda_u$  has much more chances to compensate *adverse* mistakes in the decisive pairings (those which reduce  $v_u$ ) and in the BR classifiers (so that  $v_0 > |P|$ ) because of the increased buffer of  $|N|/2$  votes from the indecisive pairings.

In a previous version of this study and in several previous publications (Loza Mencía and Fürnkranz 2010, Loza Mencía et al. 2010) we explained the underestimation with the total number of classes: when the BR classifiers included in CMLPP predict that  $\nu$  ( $= n - v_0$ ) classes are positive, then this means for the remaining classes that they have to obtain at least  $n - \nu$  votes of their maximum of  $n$  votes in order to be predicted as positive. The probability that this happens for a real positive class decreases with increasing  $n$ , since it becomes more probable that at least  $\nu$  base classifiers mistakenly take a wrong and adverse decision. This is also a possible explanation, but in retrospect the author finds the explanation given before more plausible and consistent with the empirical observations herein and in Sections 5.4.2 and 6.4.3. However, it seems worthy to further investigate this aspect in future work.

---

In order to alleviate the problem with the underestimation of the label cardinality, a simple solution entails manipulating the threshold  $v_0$  so that it is easier (or harder) for a label to be declared relevant. This parameter could be user-determined or be fitted e.g. by cross validation, as is done with several other parameterized algorithms. In order to maintain the advantage of not having to fit or set parameters, CMLPP provides the possibility of fitting the parameter (almost) *for free*. Note that for each training example, all BR classifiers  $h_{0,v}$ ,  $1 \leq v \leq n$  have to be evaluated in any case so that we obtain  $v_0$ . Moreover, in the same way, we obtain  $v_u$  for all relevant labels  $\lambda_u \in P$  and hence the corrective statistic about  $\min_u v_u - v_0$ . Correspondingly, a general, but possibly slightly more expensive approach for pairwise ensembles with incremental base learner consists in classifying training examples (or only some of them) before they are used for updating. Note that the training of the pairwise base classifiers is independent of such a bias, the training process remains untouched. Hence, a cost-efficient possibility for batch-learners is to carefully compute the bias during prediction by comparing the predicted sizes to the average prior labelset size.

#### 4.7 Summary

In this chapter, we evaluated the use of a pairwise ensemble of perceptrons for multi-label ranking and classification. Our results showed that despite the need for training a quadratic number of classifiers, the resulting incremental learning algorithm can be efficiently applied to large-scale text categorization and similar problems. We also demonstrated the effectivity of the calibration technique in order to enable the pairwise learning framework to provide a natural zero-point in the predicted label rankings.

Moreover, in terms of predictive quality, the pairwise approach compares favorably to multiclass multilabel perceptrons, an algorithm for training an one-per-class ensemble of perceptrons in a coordinated way by making the training signal of each perceptron dependent on a loss function that depends on the entire ranking, and thus dependent on the predictions of the other perceptrons in the ensemble. With the pairwise approach, we go an alternative way and try to break up the problem into independent subproblems by not trying to directly minimize any particular ranking loss, but by trying to learn the ordering relation that induces the ranking.

In contrast to the baseline, we are able to provide a set of labels as prediction in a natural way. The key idea of this approach is to introduce a calibration label that represents the boundary between relevant and irrelevant labels. We should though mention that in this case the introduction of this calibration label effectively produces an ensemble that combines the models learned by the conventional binary relevance ranking approach and those learned by the pairwise classification approach. However, the additional effort pays off, since our experimental results in the areas of text categorization, scene detection and gene analysis shows that the binary relevance approach is clearly outperformed. We have also seen some evidence that the calibration not only allows one to bipartition the ranking, but that it can also improve the quality of the ranking itself because of the increased redundancy provided by the additional classifiers in the ensemble.

---

The reason for the good performance of (C)MLPPs seems to be mainly the adequacy of this pairwise problem decomposition. Contrary to MMPs, the base perceptrons are able to find perfect classifications on the training data, which are not due to overfitting but also carry over to improved performance on the test set. Moreover, this performance does not degrade if more training epochs are used, as it does for MMPs. Thus, the problem seems to be pairwise linearly separable. We believe that this will be the case for many text categorization problems.

However, the increase in predictive performance has to be paid with a small increase in computational complexity, namely by a factor that depends on the average number of labels per example. As for most multilabel problems (in particular in text classification), this factor is rather small (about 3.24 in the *rcv1* dataset), so we consider this not to be a significant problem (cf. Section 2.9.1). The prediction time remains a problem, however we show in Chapter 5 how to considerably reduce the prediction costs to the level of training time by using Quick Voting (Park and Fürnkranz 2007) in conjunction with the artificial boundary label. But even without this optimization, the usage of the linear time perceptrons allowed to benefit from the full pairwise classification abilities on a huge dataset with more than 800,000 documents, 103 classes and 25,000 features. We believe that pairwise classification has not yet been tried with a problem of this size at the time of publication. In fact, a version with only 3000 train and test instances is frequently used (cf. provided benchmarks by Chang and Lin 2012, Tsoumakas 2012). However, an increase of the number of classes by a factor of a small one-digit number would already reach the limits of current desktop computers' memory capacity. Chapter 6 is dedicated specifically to this problem and an adapted version is presented that enables MLPP to handle thousands of classes.

Apart from the mentioned improvements, the great variety in adaptations of the perceptron algorithm allow very specific extensions of the MLPP algorithm (cf. Section 4.1.7). Note also that if the training size is small or training time and incremental learning is not important, similar benefits can be expected from using SVMs in training the base models of MLPP.



---

## 5 Efficient Aggregation Strategies

It has been shown that the complexity for training an ensemble of pairwise classifiers is comparable to the complexity of training a BR ensemble (cf. Chapter 4). The reason is that even though we have a quadratic number of classifiers in a pairwise ensemble as opposed to a linear number in the BR ensemble, each of the pairwise classifiers contains fewer examples.

However, the problem remains that a quadratic number of classifiers has to be evaluated to produce a prediction. Our first attempts in efficient multilabel pairwise classification lead to the algorithm MLPP, which uses the fast perceptron algorithm as base classifier. With this algorithm, we successfully tackled the large Reuters-rcv1 text classification benchmark, despite the quadratic number of base classifiers (cf. Chapter 4). Although we were able to beat the competing BR and the fast MMP algorithm in terms of ranking and bipartition performance and were competitive in training time, the costs for testing were not satisfactory.

Park and Fürnkranz (2007) introduced a method named *Quick Weighted Voting* (QVoting) for multiclass problems that intelligently selects only the base classifiers that are actually necessary to predict the top class. This reduced the evaluations needed from all  $n(n-1)/2$  to only  $n \log n$  in practice, which is near the  $n$  evaluations processed by BR.

In this chapter we introduce a novel algorithm which adapts the QVoting method to the MLPP algorithm. In a nutshell, the adaption works as follows: instead of stopping when the top class is determined, we repeatedly apply QVoting to the remaining classes until the final labelset is predicted. In order to determine at which position to stop, we use the calibrated label ranking technique (cf. Section 3.4.5). We evaluated this technique on a selection of multilabel datasets that vary in terms of problem domain, number of classes and label density. The results demonstrate that our modification allows the pairwise technique to process such data in comparable time to the one-per-class approaches while producing more accurate predictions.

This chapter closes the gap that was left open in the study in Chapter 4, namely the efficient prediction in the pairwise decomposition framework. Although MLPP was already able to provide a very practicable solution to large text classification problems (easily enabling real-time processing), the combination with QVoting makes a giant stride towards sufficing one of the major challenges of scalability in pairwise multilabel classification: the number of labels (cf. Section 1.1). For the first time we are able to loose the bonds of quadratic dependency inherent in the pairwise approach, namely in the phase of aggregation.<sup>45</sup>

---

<sup>45</sup> We remind that the quadratic dependency explicitly excludes the pairwise decomposition process, which may seem surprising at first sight but was already demonstrated in Section 3.4.

**Table 5.1:** Sports competition example from FIFA World Cup 2006, group F.

(a) Actual games sequence			(b) Final table		
			pos.	country	points
12 Jun	AUS : JPN	3:1	1.	Brazil	9
13 Jun	BRA : CRO	1:0	2.	Australia	4
18 Jun	BRA : AUS	2:0	3.	Croatia	2
18 Jun	JPN : CRO	0:0	4.	Japan	1
22 Jun	JPN : BRA	1:4			
22 Jun	CRO : AUS	2:2			

Nevertheless, this novel algorithm still uses a quadratic number of base classifiers, i.e. the memory requirements grow quadratically with the number of classes. Chapter 6 provides an algorithmically equivalent solution to this, although the dependency is shifted towards the number of training instances. The combination of QVoting with the hierarchical approach in Chapter 7 alleviates both factors, but it does not longer correspond to the full pairwise solution.

## 5.1 Quick Weighted Voting

As already seen, the quadratic number of base classifiers does not seem to be a serious drawback for training a (calibrated) pairwise ensemble. However, at prediction time it is still necessary to evaluate a quadratic number of base classifiers. We will sketch in the following the basic idea by means of an example from the FIFA World Cup 2006 group stage. Table 5.1(a) shows the matches in group F and Table 5.1(b) the final table.

It was clear from game three on that Brazil would at least tie at six points (at position one), since all other teams already lost one time. But only after game 5 it was obvious that Brazil would be the sole winner. For determining this, game 6 was theoretically not necessary.

A vast margin to the second team is a common observation in sports competitions. Hence, a large amount of games could often be safely omitted in order to determine the winner of a league. QVoting tries to cause such a situation as soon as possible by intelligently rearranging the games in this respect from the beginning of the competition. This is accomplished by forcing to play the team with the currently best chances of winning the whole league.

Imagine that the first game would have been Brazil vs. Croatia. QVoting would have chosen Brazil vs. Australia or Japan as next match, since Brazil had the best chances at that moment. After that win, the remaining confrontation would have determined Brazil as winner.

The principle of QVoting for multiclass and multilabel prediction is described in full detail in the following.



---

**Require:** example  $\mathbf{x}$ ; classifiers  $\{h_{u,v} \mid u < v, \lambda_u, \lambda_v \in \mathcal{L}\}; l_1, \dots, l_n = 0$

```

1: while  $\lambda_{top}$  not determined do
2:    $\lambda_u \leftarrow \arg \min_{\lambda_i \in \mathcal{L}} l_i$  ▷ select top candidate class
3:    $\lambda_v \leftarrow \arg \min_{\lambda_i \in \mathcal{L} \setminus \{\lambda_u\}} l_i$  and  $h_{u,v}$  not yet evaluated ▷ select second
4:   if no  $\lambda_v$  exists then ▷ no further unevaluated pairings with  $\lambda_u$ 
5:      $\lambda_{top} \leftarrow \lambda_u$  ▷ top rank class determined
6:   else ▷ evaluate classifier
7:      $v_{uv} \leftarrow h_{u,v}(\mathbf{x})$  ▷ one vote for  $\lambda_u$  (if  $v_{uv} = 1$ ) or  $\lambda_v$  (if  $v_{uv} = 0$ )
8:      $l_u \leftarrow l_u + (1 - v_{uv})$  ▷ update voting loss for  $\lambda_u$ 
9:      $l_v \leftarrow l_v + v_{uv}$  ▷ update voting loss for  $\lambda_v$ 

```

---

**Figure 5.1:** Pseudocode of the QVoting algorithm (multiclass classification) (based on [Park and Fürnkranz 2007](#)).

### 5.1.1 QVoting for Multiclass Classification

For the multiclass case, the simple but effective full voting strategy, which is applied often to combine the predictions of pairwise classifiers to one multiclass classification result, can be computed efficiently with the Quick Weighted Voting algorithm (*QVoting*, previously often abbreviated *QWeighting*), which is shown in Figure 5.1 ([Park and Fürnkranz 2007](#)). Instead of the evaluation of the quadratic number of all pairwise perceptrons, it is possible to evaluate a smaller subset of it in order to compute the class with the highest accumulated voting mass.

During a voting procedure there exist many situations where particular classes can be excluded from the set of possible top rank classes, even if they reach the maximal voting mass in the remaining evaluations. A more formal example as above explains the main idea: Given  $n$  classes with  $n > j$ , if class  $\lambda_u$  has received more than  $n - j$  votes and class  $\lambda_v$  lost  $j$  votings, it is impossible for  $\lambda_v$  to achieve a higher total voting mass than  $\lambda_u$ . Thus further evaluations with  $\lambda_v$  can be safely ignored for the comparison of these two classes.

Based on this intuition, [Park and Fürnkranz \(2007\)](#) developed the following algorithm: Instead of selecting the base classifiers in a predetermined order, we chose the next classifier depending on a *loss* value, which is the amount of potential voting mass that a class has *not* received. More precisely, the loss  $l_u$  of a class  $\lambda_u$  is defined as  $l_u := p_u - v_u$ , where  $p_u$  is the number of evaluated incident classifiers of  $\lambda_u$  and  $v_u$  is the current vote amount of  $\lambda_u$ . Obviously, the loss will begin with a value of zero and is monotonically increasing. The class with the current minimal loss is one of the top candidates for the top rank class.

First the pairwise classifier  $h_{u,v}$  will be selected for which the losses  $l_u$  and  $l_v$  of the relevant classes  $\lambda_u$  and  $\lambda_v$  are minimal, provided that the classifier  $h_{u,v}$  has not yet been evaluated. In the case of multiple classes that have the same minimal loss, there exists no further distinction, and we select a class randomly from this set. Then, the losses  $l_u$  and  $l_v$  will be updated based on the evaluation returned by  $h_{u,v}$  (recall that  $v_{uv}$  is interpreted

---

```

Require: example  $\mathbf{x}$ ; classifiers  $\{h_{u,v} \mid u < v, \lambda_u, \lambda_v \in \mathcal{L}\}; l_0, \dots, l_n = 0$ 
1:  $v_0 \leftarrow 0, \hat{P} \leftarrow \emptyset$ 
2:
3: for  $i = 1$  to  $n$  do                                 $\triangleright$  evaluate all classifiers of artificial label  $\lambda_0$ 
4:    $v_0 \leftarrow v_0 + h_{0,i}(\mathbf{x})$                      $\triangleright$  compute votes of calibrated label
5:
6: repeat
7:    $\lambda_{top} \leftarrow$  not determined
8:   while  $\lambda_{top}$  not determined do                     $\triangleright$  apply adapted QVoting
9:      $\lambda_u \leftarrow \arg \min_{\lambda_i \in \mathcal{L} \setminus \hat{P}} l_i$        $\triangleright$  select top candidate class
10:     $\lambda_v \leftarrow \arg \min_{\lambda_i \in \mathcal{L} \setminus \{\lambda_u\}} l_i$  and  $h_{u,v}$  not yet evaluated
11:    if  $v_u \geq v_0$  or no  $\lambda_v$  exists then             $\triangleright$  adapted stopping criterion
12:       $\lambda_{top} \leftarrow \lambda_u$ 
13:    else                                               $\triangleright$  evaluate classifier
14:       $v_{uv} \leftarrow h_{u,v}(\mathbf{x})$                      $\triangleright$  update statistics
15:       $v_u \leftarrow v_u + v_{uv}, l_u \leftarrow l_u + (1 - v_{uv})$ 
16:       $v_v \leftarrow v_v + (1 - v_{uv}), l_v \leftarrow l_v + v_{uv}$ 
17:    if  $v_{top} \geq v_0$  then
18:       $\hat{P} \leftarrow \hat{P} \cup \{\lambda_{top}\}$                  $\triangleright$  relevant label found
19:       $l_{top} \leftarrow +\infty$                          $\triangleright$  arrange  $\lambda_{top}$  at the end of possible opponents queue
20: until  $v_{top} < v_0$                                  $\triangleright$  check if all relevant labels found
21:
22: return  $\hat{P}$                                            $\triangleright$  return relevant labels

```

---

**Figure 5.2:** Pseudocode of the QCLR2 aggregation algorithm.

as the amount of the voting mass of the classifier  $h_{u,v}$  that goes to class  $\lambda_u$  and  $1 - v_{uv}$  is the amount that goes to class  $\lambda_v$ ). These two steps will be repeated until all classifiers for the class  $\lambda_u$  with the minimal loss has been evaluated. Thus the current loss  $l_u$  is the correct loss for this class. As all other classes already have a greater loss,  $\lambda_u$  is the correct *top rank* class.

Theoretically, a minimal number of comparisons of  $n - 1$  is possible (*best case*) if the top class is selected first and it is correctly preferred in all comparisons (Park and Fürnkranz 2007). The *worst case*, on the other hand, is still  $n(n - 1)/2$  comparisons, which can, e.g., occur if all pairwise classifiers classify randomly with a probability of 0.5. In practice, the number of comparisons will be somewhere between these two extremes, depending on the nature of the problem. Section 5.2 analyses the costs in more detail.

### 5.1.2 QVoting for Multilabel Classification

A simple adaptation of QVoting to multilabel classification consists in repeating the process. We can compute the top class  $\lambda_{top}$  using QVoting, remove this class from  $\mathcal{L}$  and

---

repeat this step until the returned class is the artificial label  $\lambda_0$ , which means that all remaining classes will be considered to be irrelevant. This adaptation uses two simple extensions of the original algorithm. Firstly, the information about which pairwise classifiers have been evaluated and their results are carried through the iterations so that no base classifier is evaluated more than once. And secondly, by using the calibrated label ranking approach we know beforehand that at some point the vote amount of the artificial label has to be computed since this is the number of votes a label must at least obtain. So, in hope for a *better* starting distribution of votes, all incident classifiers  $h_{i,0}$  of the artificial label are evaluated explicitly before employing iterated QVoting. We refer to this method as *QCLR1* and *QCM LPP1* if perceptrons are used as base classifiers (cf. Chapter 4).

In addition to this straightforward adaptation, we considered also a slightly improved variant (*QCLR2* or *QCM LPP2* respectively if MLPP is used). In retrospect, *QCLR1* computes a partial ranking of classes down to the calibrated label. That means that for all relevant labels all their incident classifiers are evaluated. It neglects the fact that for multilabel classification the information that a particular class is *ranked above* the calibrated label is sufficient, rather than *to which amount*. *QCLR2* works in the same way as *QCLR1* except that it stops the evaluation of the current top rank  $\lambda_{top}$  if it already received a higher voting mass than the calibrated label. The class  $\lambda_{top}$  is not automatically removed from the set of labels as in *QCLR1*, since further evaluations for the computation of other classes can occur, but it can not be selected as a new top rank candidate. In addition, it is arranged at the end of possible opponents, since a match against  $\mathcal{L} \setminus \hat{P}$  will help the probably relevant  $\lambda_u$  more than a random outcome from a probably *incompetent* classifier  $h_{u,v}$ ,  $\lambda_v \in \hat{P}$  (cf. Section 3.5.2).

Note that the effectiveness of this testing procedure is highly dependent on the relation of average number of relevant labels to total number of labels. We can expect a high reduction of pairwise comparisons if the above relation is relatively small, which holds for the most real-world multilabel datasets (cf. Table 2.2).

### 5.1.3 Extensions

Different search heuristics based on other losses than the number of “lost games” are imaginable. Furthermore, the selection of the two next classes for evaluation can also be varied, i.e. by pairing the “best” and the “worst” class in the next iteration instead of the two currently best classes. In the same way, and especially in the beginning, using the apriori label probabilities instead of randomly undoing the ties could lead to a further improvement.

QVoting has been recently extended to a general framework supporting a great variety of different types of multiclass decompositions (Park and Fürnkranz 2012), specifically any decomposition representable as ternary ECOC matrix (cf. Section 3.3).

## 5.2 Computational Complexity

The presented approaches do not change the training or the storage of the pairwise ensemble, hence it remains quadratic with respect to the number of labels as indicated in Section 3.4.6 and Table 3.2. The prediction has the following reduced computational costs given in number of classifier evaluations and in dependency of number of possible classes  $n$  and the average number of relevant classes per instance  $d$  in the training set.

We begin with an analysis of the base multiclass QVoting approach. We idealistically assume perfectly classifying base classifiers, i.e.  $P(h_{u,v}(\mathbf{x}) = y_u \mid y_u + y_v = 1) = 1$  and  $P(h_{u,v}(\mathbf{x}) = 1 \mid y_u + y_v \neq 1) = 1/2$ . Once the true  $\lambda_{top} \in P$  is selected as candidate  $\lambda_u$  or  $\lambda_v$ , it is also chosen the next  $n-2$  times until it is determined as the winner and the algorithm ends. In the best case  $\lambda_{top}$  is selected randomly as the first candidate  $\lambda_u$ , leading to in total  $n-1$  base classifier evaluations. The worst case corresponds to the  $n-1$ -th position and in total  $2n-3$  evaluations. We can hence determine  $n-1$  as absolute and  $\frac{3}{2}n-2$  as asymptotical lower bound for computing QVoting.

In the real world we have to relax our idealistic assumptions regarding the error rate of the competent classifiers and the bias of the incompetent classifiers. Certainly, the absolute upper bound of evaluations is still  $n(n-1)/2$ , e.g. when  $P(h_{u,v} = 1) = 1/2, \forall \lambda_u, \lambda_v \in \mathcal{L}$ . But fortunately, the number of comparisons will be somewhere between these two extremes in practice, depending on the nature of the problem. Previous experiments of [Park and Fürnkranz \(2007\)](#) have shown for the multiclass case that QVoting reduces the amount of required base classifier evaluations from  $\frac{n(n-1)}{2}$  to approximately  $n \log(n)$  in practice. Hence, we will use this as reference for the analysis of the combined QCLR approaches.

Let  $C_{QV}$  be the runtime of one iteration of QVoting. Then, it is easy to see that the number of base classifier evaluations for the multilabel adaptations of QVoting is bounded from above by  $n + |P| \cdot C_{QV}$ , since we always evaluate the  $n$  classifiers involving the calibrated class, and have to do one iteration of QVoting for each of the  $|P|$  relevant labels. Assuming that QVoting on average needs  $C_{QV} = n \log(n)$  base classifier evaluations we can expect an average number of  $n + dn \log n$  classifier evaluations for the QCLR variants, as compared to the  $\approx n^2$  evaluations for the regular pairwise testing. Thus, the effectiveness of the adaption to the multilabel case crucially depends on the average number  $d$  of relevant labels. We can expect a high reduction of pairwise comparisons if  $d$  is small compared to  $n$ , which holds for most real-world multilabel datasets (cf. Section 2.9.1). Moreover, there is no disadvantage of using QCLR instead of the normal aggregation process unless a more fine-grained distinction between classes than relevant-irrelevant is required.

A compilation of the analysis can be found in Table 5.2, together with the complexities of BR. Note that the stated prediction time for QCLR in the table is not an analytical complexity bound like the others, but it is an empirically estimated asymptotic value.

**Table 5.2:** Computational complexity comparison of the QVoting approach.  $n$  denotes the total number of classes and  $d$  the average cardinality of the labelsets.

	trained classifiers in memory	classifier evaluations during prediction
BR	$n$	$n$
CLR	$\frac{n(n+1)}{2}$	$\frac{n(n+1)}{2}$
QCLR	$\frac{n(n+1)}{2}$	$\sim n + dn \log n$

### 5.3 Experimental Setup

The next subsections describe the experimental setup for the experiments which compare CMLPP with its optimized variant QCMLPP. The additional experiments with SVMs as base classifiers, the setup and the results are presented in Section 5.4.3.

#### 5.3.1 Datasets

The datasets that were included in the experimental setup cover three application areas in which multilabeled data are frequently observed: *text categorization* (among others, Reuters *rcv1* and *r21578* and the *EUR-Lex* dataset), *multimedia classification* (the *scene*, *mediamill* and *emotions* datasets) and *bioinformatics* (*yeast* and *genbase*). Table 5.3 provides an overview of the different relevant characteristics of the used datasets, which were already presented in Section 2.9.1.

Both Reuters test beds were preprocessed as indicated in Section 2.9.1, i.e. 25.000 text features remained for *rcv1* and 10.000 for *r21578*. The *EUR-Lex* datasets will be extensively presented in Section 6.1. In these experiments we focus on the *subject matter* (*eurlex\_sm*) and *directory code* (*eurlex\_dc*) subsets since the last one *EUROVOC* with 3956 classes would require to maintain almost 8 million classifiers in memory which is only feasible with the techniques introduced in Section 6.2. 5.000 features have been selected.

In the previous experiments with MLPP on *yeast* and *scene* presented in Chapter 4, we found that even the pairwise problems are hard to separate with a linear classifier (much more so in the binary relevance setting). Thus, in this set of experiments, we added all pairwise feature products to the original feature representation, in order to simulate a quadratic kernel function.

#### 5.3.2 Algorithmic Setup

All algorithms are trained incrementally. For the *rcv1* dataset, a single, chronological pass through the data was used (one epoch) because our previous results have shown that

**Table 5.3:** Statistics of datasets. The attribute number in parenthesis refer to the actual used number of features, i.e. for *scene* and *yeast* the number of features after adding the pairwise products and for the text collections the amount after feature selection. *Labelset size*  $d$  denotes the average number of labels per instance, and *label density* indicates the average number of labels per instance  $d$  relative to the total number of classes  $n$ .

dataset	$n$	#instances $m$	#attributes $a$	$d$	density
<i>scene</i>	6	2407	294 (86732)	1.07	17.9 %
<i>emotions</i>	6	593	72	1.87	31.1 %
<i>yeast</i>	14	2417	103 (10712)	4.24	30.3 %
<i>tmc2007</i>	22	28596	49060	2.16	9.8 %
<i>genbase</i>	27	662	1186	1.25	4.6 %
<i>medical</i>	45	978	1449	1.25	2.8 %
<i>enron</i>	53	1702	1001	3.39	6.4 %
<i>mediamill</i>	101	43907	120	4.38	4.3 %
<i>rcv1</i>	101	804414	231188 (25000)	3.24	3.1 %
<i>r21578</i>	120	11367	21474 (10000)	1.26	1.0 %
<i>bibtex</i>	159	7395	1836	2.4	1.5 %
<i>eurlex_sm</i>	201	19348	166448 (5000)	2.21	1.1 %
<i>eurlex_dc</i>	410	19348	166448 (5000)	1.29	0.3 %
<i>delicious</i>	983	16105	500	19.02	1.9 %

multiple iterations are not necessary (cf. Chapter 4). For the remaining text classification tasks we report the results for 10 epochs. The classifiers for the supposedly more difficult non-textual datasets were trained using 100 epochs. However, in terms of the relative order of the tested methods, we found that the results are quite insensitive to the exact numbers of epochs.

For *yeast*, *scene*, *r21578* and *EUR-Lex* the reported results are estimated from 10-fold cross-validation (cf. Section 2.7.2). In order to ensure that no information from the test set enters the training phase for the text datasets, the TF-IDF transformation and the feature selection were conducted only on the training sets of the cross-validation splits. For datasets for which it was not indicated we used the first two-thirds of examples for training and the remaining for testing. Particularly, we used 391 training examples for *emotions*, 21519 for *tmc2007*, 463 for *genbase*, 465 for *medical*, 1123 for *enron*, 30993 for *mediamill*, the aforementioned 535,987 for *rcv*, 4930 documents for *bibtex* and 12,920 for *delicious*.

We initialized the perceptrons in MLPP with random values.

## 5.4 Evaluation

The following sections analyze, in short, the predictive quality and in a more extensive way the computational efficiency of the presented algorithms.

**Table 5.4:** Computational costs at prediction in average number of classifier evaluations per instance. The italic values next to the two multilabel adaptations of QVoting show the ratio of classifier evaluations to CMLPP and the second rightmost column describes the average number of relevant labels.

dataset	$n$	BR	CMLPP	QCMLPP1	QCMLPP2	$n \log(n)$	$n + dn \log(n)$	$d$	density $\frac{d}{n}$
scene	6	6	21	11.51 (54.8%)	11.46 (54.6%)	10.75	17.50	1.07	17.9 %
emotions	6	6	21	17.03 (81.1%)	16.59 (79.0%)	10.75	26.10	1.87	31.2 %
yeast	14	14	105	67.57 (64.4%)	64.99 (61.9%)	36.94	170.65	4.24	30.3 %
tmc2007	22	22	253	81.76 (32.3%)	78.01 (30.8%)	68.00	168.89	2.16	9.82 %
genbase	27	27	378	71.53 (18.9%)	62.11 (16.4%)	88.99	138.23	1.25	4.63 %
medical	45	45	1035	112.78 (10.9%)	103.67 (10.0%)	171.30	259.12	1.25	2.78 %
enron	53	53	1431	286.36 (20.0%)	262.30 (18.3%)	210.43	764.24	3.38	6.38 %
mediamill	101	101	5151	489.45 (9.50%)	378.04 (7.34%)	466.13	2142.64	4.38	4.34 %
rcv1	103	103	5356	485.23 (9.06%)	456.23 (8.52%)	477.38	1649.70	3.24	3.15 %
r21578	120	120	7260	378.45 (5.21%)	325.94 (4.49%)	574.50	843.87	1.26	1.05 %
bibtex	159	159	12720	604.37 (4.75%)	492.73 (3.87%)	805.96	2093.29	2.40	1.51 %
eurlex_sm	201	201	20301	926.71 (4.56%)	771.62 (3.80%)	1065.96	2556.78	2.21	1.10 %
eurlex_dc	410	410	83845	1667.16 (1.98%)	1136.85 (1.35%)	2466.62	3591.95	1.29	0.31 %
delicious	983	983	483636	48680.12 (10.1%)	46835.89 (9.68%)	6773.47	129814.40	19.02	1.93 %



---

### 5.4.1 Computational Efficiency

Our analysis of computational efficiency concentrates on the savings in base classifier evaluations using the QVoting method on the different multilabel datasets.

Table 5.4 depicts the gained reduction of prediction complexity of the QVoting approach with respect to the classifier evaluations for CMLPP. For each of the four listed methods (BR, CMLPP, QCMLPP1 and QCMLPP2) the average number of base classifier evaluations is stated. In addition, for QCMLPP1 and 2 the ratio of classifier evaluations to the complete set of pairwise classifiers, which are typically evaluated in the CMLPP approach, are denoted within brackets, to emphasize the achieved reduction.

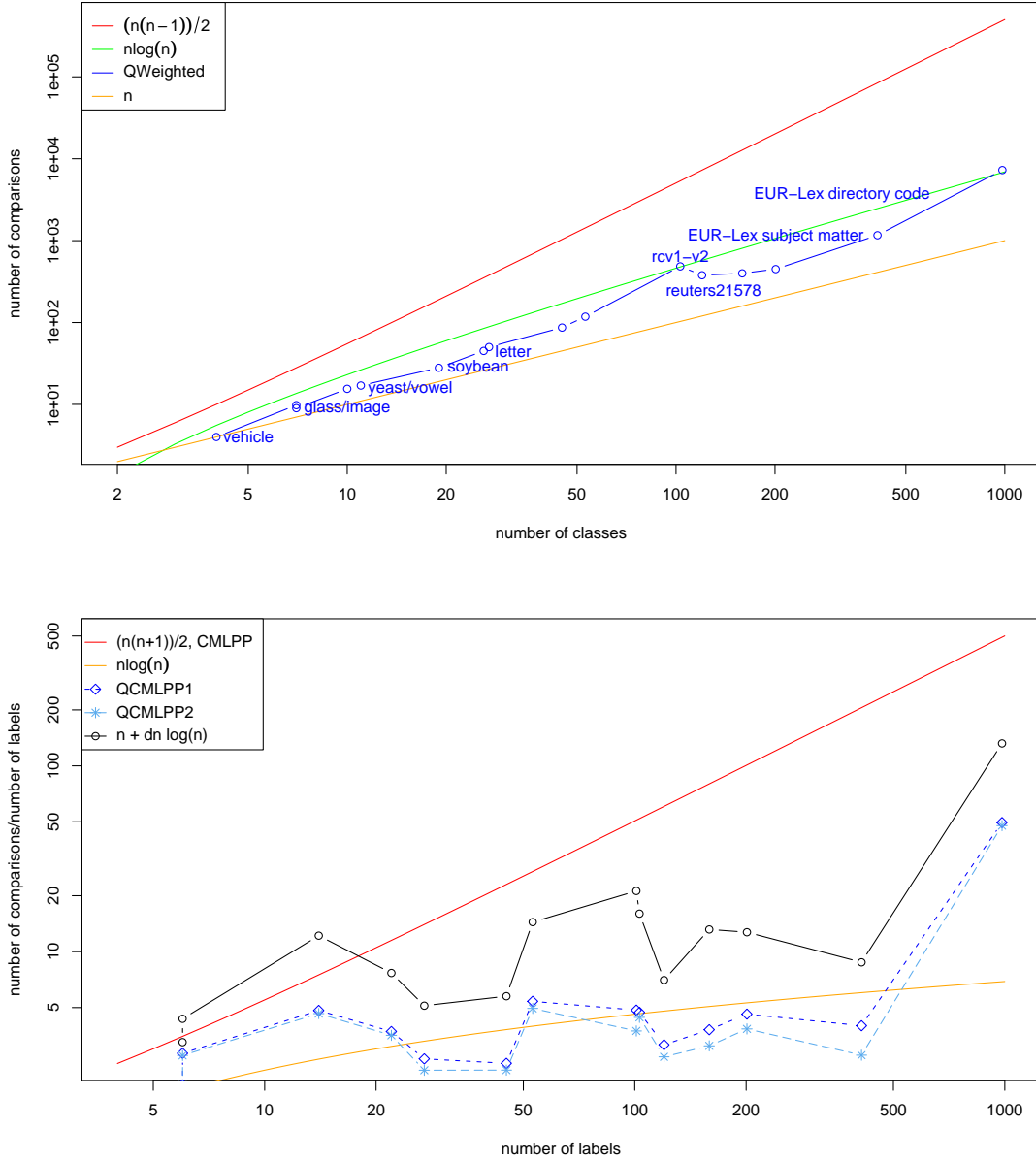
The first remarkable observation is the clear improvement when using the QVoting approach. Except for the four smallest datasets regarding the labelsize, both variants of the QCMLPP use less than 20 percent of the classifier evaluations for CMLPP.

Another appreciable point, especially regarding the mentioned deviation, is the clearly visible correlation between the gained reduction and the label density of the problem, i.e. the ratio of the average number of labels per instance to the total number of labels. The dataset with the highest density, *emotions*, achieved the lowest reduction, followed by *yeast* with a similar density and reduction ratio. Similarly, both QCMLPP variants evaluated the lowest ratio of classifiers for the dataset with the lowest density, the *eurllex\_dc* dataset. This observation confirms the previously stated expectation that the reduction is highly influenced by the density. This effect is not surprising, since roughly speaking QCMLPP employs iteratively QVoting until the calibrated label is found, and the number of iterations is obviously related to the density. Furthermore the results show that QCMLPP2 slightly but constantly outperforms QCMLPP1.

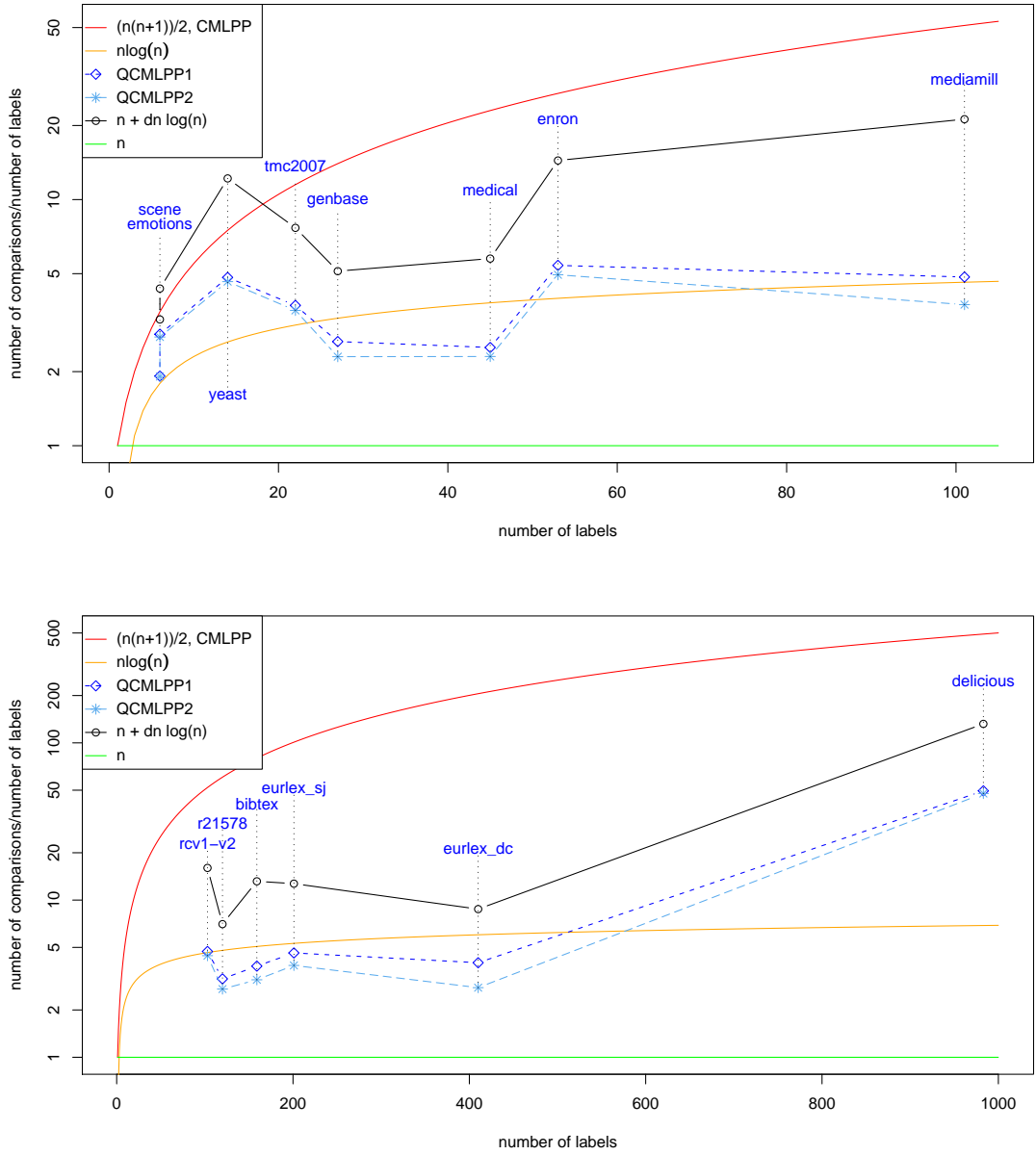
For estimating the average runtime in practice, two columns were included, which state the  $n \log(n)$  and  $n + dn \log(n)$  values for the corresponding datasets. We can clearly confirm that the number of classifier evaluations is for all considered datasets smaller than the previously estimated upper bound of  $n + dn \log(n)$ . Note that the value for *yeast* (170.65) is actually greater than the number of existing classifiers (105). This is due to the fact that the values lie yet in a range where lower order terms have still an impact in the equation.

Figure 5.3 visualizes the above results and allows again a comparison to different complexity values such as  $n$ ,  $n \log(n)$  and  $n^2$ . The upper figure is a recapitulation of the results from Park and Fürnkranz (2007) extended with multiclass classification performance results of the multilabel datasets considered in this paper: instead of evaluating until finding the calibrating label, QVoting was only applied once such as if it was a multiclass problem. These results for the simulated multiclass classification performance support additionally the statement that QVoting achieves an  $n \log(n)$  runtime in practice. For better readability, a logarithmic scale for both axis is used. The lower figure is more interesting in this context, where multilabel classification prediction complexity of QCMLPP is presented. Note that the y-axis now describes the number of comparisons (or classifier evaluations, respectively) divided by the number of labels, which is graphically motivated and allows a finer distinction of the different curves. Note also that for the





**Figure 5.3:** Prediction complexity of QVoting and QCMLPP: number of comparisons needed in dependency of the number of classes  $n$  for different multiclass and multilabel problems. *Upper figure:* Problems *vehicle* to *letter* in the first figure are multiclass problems already analyzed by [Park and Fürnkranz \(2007\)](#), while multiclass versions of the multilabel datasets as described in Table 5.3 were evaluated within this study. *Lower figure:* QCMLPP1/2 is compared to  $n(n + 1)/2$  as in CMLPP,  $n$  as in BR and  $n \log(n)$  on 14 multilabel datasets.



**Figure 5.4:** Prediction complexity of QCMLPP: the upper figure contains the small datasets, the bottom figure the large datasets.

black curve ( $n + dn \log(n)$ ) the actual average number of labels from the data was used for computing the values and are identical to the ones from Table 5.4. These values are also depicted in the additional Figure 5.4, which shows again the comparison of computational costs split into two figures, the first for smaller datasets with  $n < 103$  and the second for larger datasets. In comparison to Figure 5.3, the x-axis is now linear and we have added the dataset names to the data points.

As we can see from these figures, the empirical runtime bound  $n + dn \log(n)$  is never exceeded. We conclude that this estimate is a reasonable indicator for the runtime complexity of QCMLPP.

#### 5.4.2 Predictive Quality

Although it is not the focus of this study, we will compare in this section the prediction quality of BR and CMLPP in order to demonstrate the expected advantage of the pairwise approach. Note that the multilabel losses of the QCMLPP are exactly equal to those of CMLPP since both compute for every instance the same partitioning into relevant and irrelevant labels. Table 5.5 shows the label set predictions performance according to Section 2.7.3.<sup>46</sup> Four of the datasets were already used for evaluating CMLPP in Section 4.6. But note that even though QVoting allowed to extend the empirical basis for evaluating CMLPP due to the improvement in testing efficiency, QCLR does not allow a useful comparison and analysis of the ranking performance since it is not interested in providing a good ranking: after it is clear which labels are at top of the calibrated label, no additional computations are done in order to determine the remaining ordering. Ranking losses produced by CMLPP on *rcv1*, *r21579*, *scene* and *yeast* were already presented in Section 4.6. We also refer the interested reader to Loza Mencía et al. (2008, Table 7.1) for a comparison on ranking losses including QCMLPP.

The first remarkable observation is that for the overall evaluation measures HAMLoss and F1 the pairwise approach dominates the one-per-class approach for every dataset except *genbase* and *medical*. BR's PREC is outperformed on all datasets.

On the other hand, QCMLPP achieves a lower REC for the datasets with slightly more than 100 classes, beginning at *rcv1* with 103 classes, and *genbase* and *medical*. As already outlined in Section 4.6.9, this is probably due to the fact that calibration tends to underestimate the number of returned labels for each instance, especially for datasets with a high density. The reason is that with increasing density the relevant labels can obtain more votes from *indecisive* classifiers which they would normally not receive, and hence they can more easily equalize missing votes from mistakes in the *decisive* classifiers. Note that the datasets with a higher recall for CMLPP have densities in the interval between 3.1% and 31.1% and the others only densities between 0.3% and 4.6%, confirming our explanation attempt. The corrective measure presented in Section 4.6.9 that adapts  $v_0$  by subtracting a bias is perfectly applicable to QCMLPP.

<sup>46</sup> The results for *rcv1* and *r21578* slightly differ from those of Loza Mencía et al. (2010) since the values were adopted from the repeated experiments in Section 4.6 in order to be consistent.

**Table 5.5:** Bipartitioning performance of the different algorithms. For HAMLoss low values are good, for the other three measures the higher the better. Bold values represent the best value for each dataset and measure combination. Note that the multilabel losses of QCMLPP are exactly equal to those of CMLPP.

dataset	$n$	HAMLoss		PREC		REC		F1	
		BR	CMLPP	BR	CMLPP	BR	CMLPP	BR	CMLPP
<i>scene</i>	6	10.42	<b>10.00</b>	71.80	<b>71.83</b>	71.21	<b>74.20</b>	71.19	<b>72.76</b>
<i>emotions</i>	6	35.64	<b>34.08</b>	46.78	<b>48.62</b>	60.15	<b>61.90</b>	52.63	<b>54.47</b>
<i>yeast</i>	14	24.09	<b>22.67</b>	60.47	<b>62.37</b>	59.07	<b>63.31</b>	59.76	<b>62.83</b>
<i>tmc2007</i>	22	7.37	<b>6.78</b>	62.57	<b>64.16</b>	66.47	<b>73.61</b>	64.46	<b>68.56</b>
<i>genbase</i>	27	<b>0.26</b>	0.48	99.22	<b>99.59</b>	<b>95.49</b>	90.60	<b>97.32</b>	94.88
<i>medical</i>	45	<b>1.51</b>	1.51	71.72	<b>76.02</b>	<b>75.84</b>	66.75	<b>73.72</b>	71.08
<i>enron</i>	53	7.56	<b>6.01</b>	41.56	<b>52.82</b>	47.05	<b>49.51</b>	44.13	<b>51.11</b>
<i>mediamill</i>	101	4.52	<b>4.16</b>	42.28	<b>56.66</b>	10.05	<b>19.70</b>	16.24	<b>29.23</b>
<i>rcv1</i>	103	1.26	<b>1.03</b>	80.15	<b>86.77</b>	<b>79.70</b>	79.33	79.93	<b>82.88</b>
<i>r21578</i>	120	0.40	<b>0.29</b>	78.38	<b>87.98</b>	<b>85.59</b>	83.79	81.81	<b>85.82</b>
<i>bibtex</i>	159	1.57	<b>1.35</b>	46.53	<b>57.97</b>	<b>36.30</b>	34.84	40.78	<b>43.53</b>
<i>eurlex_sm</i>	201	0.76	<b>0.54</b>	63.39	<b>77.88</b>	<b>74.11</b>	71.57	68.32	<b>74.59</b>
<i>eurlex_dc</i>	410	0.26	<b>0.17</b>	56.26	<b>79.21</b>	<b>70.54</b>	61.98	62.58	<b>69.54</b>
<i>delicious</i>	983	5.58	<b>3.48</b>	11.88	<b>19.77</b>	<b>29.59</b>	26.51	16.95	<b>22.65</b>

However, a look at the average predicted labelset size shows that an underestimation only occurs for the *EUR-Lex* datasets and not for *r21578* or *delicious* ( $n \geq 120$ ). For *delicious* QCMLPP even predicts more than 25 instead of 19 labels. On the other hand we can observe that BR always predicted a higher label number than QCMLPP on the datasets in which it achieved a higher REC. One extreme are the 47 predicted labels for *delicious*, but note that in general it cannot be stated that BR overestimates the number of labels.

Take also into consideration that it is easily possible to bias the recall/precision trade-off of the calibration by simply subtracting or adding a fixed number of votes to the artificial class count.

### 5.4.3 Support Vector Machines

We conducted experiments with support vector machines as base learners in order to demonstrate that the same positive effects can also be expected from the pairwise approach and the QVoting optimization when using a different base learner. We used the LibSVM implementation (Chang and Lin 2001) with standard settings, which uses the RDF kernel, for the non-textual datasets and the efficient LibLINEAR implementation (Fan et al. 2008) for textual datasets with the primal L2-loss SVM option, which is supposed to enhance speed (Hsu et al. 2009a). We ignored the results on *genbase* since

**Table 5.6:** SVM as base learner – Computational costs at prediction in average number of classifier evaluations per instance. The italic values next to the multilabel adaptation of QVoting (QCLR2) shows the ratio of classifier evaluations to CLR and the second rightmost column describes the average number of relevant labels.

dataset	$n$	BR	CLR	QCLR2	$n \log(n)$	$n + dn \log(n)$	$d$
scene	6	6	21	7.88 (37.5%)	10.75	17.50	1.07
emotions	6	6	21	11.87 (56.5%)	10.75	26.10	1.87
yeast	14	14	105	40.31 (38.4%)	36.94	170.65	4.24
tmc2007	22	22	253	68.92 (27.2%)	68.00	168.89	2.16
medical	45	45	1035	97.40 (9.41%)	171.30	259.12	1.25
enron	53	53	1431	223.42 (15.6%)	210.43	764.24	3.38
r21578	120	120	7260	303.90 (4.19%)	574.50	843.87	1.26
bibtex	159	159	12720	485.97 (3.82%)	805.96	2093.29	2.40

LibSVM predicted the empty label set on all test examples. For the remaining missing datasets no results could be retrieved due to the higher memory requirements of the SVMs compared to the simple perceptrons. For *yeast* and *scene* we did not use the quadratic kernel simulation anymore.

Table 5.6 shows the computational costs of QCLR2 with SVM as base classifier. We can observe an overall similar picture compared to the results of Table 5.4: the pairwise approach clearly benefits from the QVoting optimization. However, while the reduction in number of required comparisons for the textual datasets is very similar, using LibSVM seems to allow to further improve the ratio on the non-textual *scene*, *emotions* and *yeast*. The explanation can be seen in Table 5.7, which lists the prediction quality for BR and QCLR2. A very high precision is achieved by LibSVM for these datasets due to predicting only a small number of labels. This cautious behavior of LibSVM could already be observed for the *genbase* dataset. QCMLPP2 with perceptrons as base classifier e.g. predicts 2.51 labels on average on the *emotions* test set, while with SVM as base classifier only 1.27 are predicted. This means for QCLR on average more than one additional QVoting iteration for each example during classification, which is the reason for the further reduction of the computational costs. An additional iteration can easily automatically be enforced by biasing the stopping criterion as shown in Section 4.6.9.

Note that although the obtained reductions in number of base classifiers is similar for both perceptrons and SVM, training the SVMs does usually require a higher amount of CPU-time. Except for *emotions*, for which the time is almost equal, and *yeast* and *scene*, which are not directly comparable due to the different feature representations used, the perceptrons are always faster, namely  $2.3\times$  faster for *tmc2007* to even  $29\times$  faster for *enron*.

Especially if we consider that the prediction quality of perceptrons and SVMs are very similar (at least for the text classification tasks), this constitutes an important point in defense of the perceptron algorithm. However, it is also interesting to observe that the

**Table 5.7:** Bipartitioning performance of the different algorithms with SVM as base learner. For HAMLoss low values are good, for the other three measures the higher the better. Bold values represent the best value for each dataset and measure combination. Note that the multilabel losses of QCLR are exactly equal to those of CLR.

dataset	$n$	HAMLoss		PREC		REC		F1	
		BR	CLR	BR	CLR	BR	CLR	BR	CLR
<i>scene</i>	6	12.57	<b>12.51</b>	<b>93.25</b>	93.04	32.16	<b>32.58</b>	47.77	<b>48.21</b>
<i>emotions</i>	6	27.56	<b>26.57</b>	<b>65.55</b>	64.98	34.34	<b>41.85</b>	45.07	<b>50.91</b>
<i>yeast</i>	14	22.51	22.51	<b>75.61</b>	75.60	37.81	<b>37.82</b>	50.41	<b>50.41</b>
<i>tmc2007</i>	22	6.99	<b>6.63</b>	66.16	<b>67.31</b>	62.33	<b>66.16</b>	64.19	<b>66.73</b>
<i>medical</i>	45	<b>1.09</b>	1.11	<b>83.12</b>	82.10	76.56	<b>76.79</b>	<b>79.70</b>	79.36
<i>enron</i>	53	5.70	<b>5.22</b>	55.87	<b>59.95</b>	48.64	<b>53.36</b>	52.00	<b>56.47</b>
<i>r21578</i>	120	0.56	<b>0.55</b>	71.23	<b>71.76</b>	<b>78.49</b>	78.34	74.68	<b>74.90</b>
<i>bibtex</i>	159	1.48	<b>1.39</b>	50.45	<b>54.65</b>	37.60	<b>39.32</b>	43.09	<b>45.73</b>

distance between BR and QCLR is considerably reduced when using SVMs, which might be an indication for a higher robustness of the pairwise approach against weak base classifiers.

Following the discussions about using perceptrons or SVMs in Section 4.1.7 and 4.5.1, it would be interesting to make a comparison on basis on the reported results in Table 5.5 and 5.6. Unfortunately, a direct comparison is difficult on basis of bipartitioning results due to the different tradeoff between recall and precision. For instance, QCMLPP clearly obtains a higher overall F1 on the smaller datasets, but this is only due to the relatively high precision and low recall of the SVM variant. A matching or similar recall or precision between the perceptron or SVM based methods would facilitate a reasonable comparison, but this is also not given for the set of larger datasets. However, recall and precision are more outbalanced for SVM on these datasets and hence we can observe a slight advantage over the perceptrons with respect to F1.

## 5.5 Discussion and Related Work

Very recently, Madjarov et al. (2012) presented an extension of QCLR based on the clear separation of one-against-all and pairwise base classifiers. Only the prediction phase is affected, the trained ensemble is exactly the same as in (Q)CLR. In a first stage and similarly to QCLR, the BR classifiers  $h_{u,0}$  are consulted in order to predict a ranking on the labels. According to a threshold, which is determined via cross validation, a separation in relevant  $\hat{P}$  and irrelevant labels  $\hat{N}$  is obtained and is maintained until the end. If the threshold was zero like in (Q)CLR and conventional BR, their Two Stage Voting Method (TSVM) would hence produce the same bipartition predictions as BR. In the sec-

ond phase, a voting is performed in order to obtain a ranking on  $\hat{P}$ . The considered votes are determined from querying the classifiers  $h_{u,v}, \lambda_u, \lambda_v \in \hat{P}$ .<sup>47</sup>

Not surprisingly, the reduction in BC evaluations is higher than for QCLR since the number amounts to  $\mathcal{O}(n + d(d - 1)/2)$ . But especially on datasets with a high label dimensionality, the savings relation to CLR is often comparable. E.g. on *enron*, TSVM achieves a reduction to 11.7% of the comparisons whereas QCLR2 needs 15.6% and QCMLPP 18.2% (cf. their Table 8, column  $r_{real}$ -TSVM). The predictive quality is comparable to CLR, though especially for big datasets we can observe a clear tendency of better ranking ability of CLR, also reflected in the diagrams which show the quality in dependency of  $|\hat{P}|$ , i.e. the number of pairwise comparisons. Their experiments also confirm the superiority of the pairwise approaches compared to BR, label powerset (cf. Section 3.2) (significant in many cases) and also classifier chains (cf. Section 3.5.7). Based on the last mentioned stacking classifier chains (CC), Madjarov et al. also present two enhancements of TSVM which incorporate the predictions, or more specifically the training signals, of all BR classifiers (TSCCM) or only the affected ones ( $h_{u,0}$  and  $h_{v,0}$  for  $h_{u,v}$ , TSPCCM) as additional features for the training instances. Especially the last method deserves to be considered in future works since it does almost not increase the computational costs and it was able to slightly increase the performance in their two-stage method.

Another interesting point is their approach to predict a ranking on  $\tilde{N}$ . Madjarov et al. simply carry over the scores of the BR classifiers and this seems to be enough in order to obtain at least competitive rankings to CLR.<sup>48</sup> Hence, this seems to be an appropriate approach in order to enable QCLR to predict reasonable rankings in addition to bipartitions.

## 5.6 Summary

The main disadvantage of the approach of learning by pairwise comparison was, until now, the quadratic number of base classifiers needed and hence the increased computational costs for prediction and the increased memory requirements. We have presented in this chapter a time efficient algorithm for accelerating the aggregation process without altering the prediction itself.

The proposed approach combines the calibration technique that transforms a class ranking into a bipartite prediction by introducing an artificial thresholding class (cf. Section 3.4.5) with the Quick Weighted Voting (Park and Fürnkranz 2007) that stops the computation of the ranking when the bipartite separation is already determined .

<sup>47</sup> In their explanation they statically add the votes resulting from  $\hat{P} \succ_{BR} \tilde{N}$ . But since every class in  $\hat{P}$  would receive the same amount  $|\tilde{N}|$  of positive votes, this step can be omitted.

<sup>48</sup> In one paragraph (p. 1032) they contradict and say that the labels in  $\tilde{N}$  are ranked according to the votes, but from the previous descriptions and the results it seems obvious that this is not the case. Another discrepancy is that QCLR and CLR perform differently on Hamming loss. They employed our implementation of QCLR in Mulan (Tsoumakas et al. 2011b) for experimentation, which definitely produced the same bipartitions as CLR in our set of experiments.

---

For the combined QVoting multilabel method the computational costs savings compared to the normal voting are especially important with increasing number of classes. Though not analytically proven, our empirical results show that the complexity is upper bounded by  $n + dn \log(n)$ , in comparison to the evaluation of  $n$  in the case of the binary relevance approach and  $n(n + 1)/2$  for the unmodified pairwise approach.

The benefit in predictive quality of using CMLPP against using BR was shown by an extensive experimental evaluation on 14 datasets. Together with QVoting, CMLPP is able to achieve a good trade-off between predictive quality and speed. Additional experiments using state-of-the-art support vector machines as base learner instead of the perceptron algorithm confirmed that the binary relevance approach is outperformed by the pairwise approach. These experiments also show that the advantage of using the pairwise approach and QVoting is independent of the base learner employed. In addition, we could observe indications for a increased robustness against weak base classifiers for the pairwise ensemble.

The key remaining bottleneck is that we still need to store a quadratic number of base classifiers, because each of them may be relevant for some example. The extension to MLPP presented in the next section is mainly dedicated to this issue. The combination with HOMER in Chapter 7, an algorithm that arranges multilabel base classifiers in a hierarchical tree, also achieves to reduce memory consumption (in addition to computational costs). But it is no longer equivalent to the original pairwise decomposition in which each possible pairwise relation between labels is modeled, while QVoting aggregation and also the MLPP based extension mentioned above explicitly are.



---

## 6 Highly Scalable Dual Models

A very important challenge for multilabel classification, and certainly the most important one for pairwise decomposition, is high label dimensionality (cf. Section 1.1). Section 4.4 introduced the perceptron based MLPP algorithm, which demonstrated its suitability on demanding datasets such as Reuters with more than 100 labels. In Chapter 6 a dataset with even almost thousand classes appeared and was successfully tackled by MLPP, but this was only possible due to the extreme reduction (already in the original source) of the feature space to 500 dimensions. But if such a limiting reduction is not desired and if the label dimensionality increases to even higher levels, the limits of pairwise decomposition scalability are surely exceeded. The present chapter introduces an adaptation based on the internals of the perceptron algorithm that makes problems with these extreme characteristics attainable to pairwise decomposition. The starting point of this study is the following very concrete multilabel text classification challenge.

The *EUR-Lex* text collection is a collection of documents about European Union law. It contains many different types of documents, including treaties, legislation, case-law and legislative proposals, which are indexed according to several orthogonal categorization schemes to allow for multiple search facilities. The most important categorization is provided by the *EUROVOC descriptors*, which is a topic hierarchy with almost 4000 categories regarding different aspects of European law.

This document collection provides an excellent opportunity to study text classification techniques for several reasons:

- it contains multiple classifications of the same documents, making it possible to analyze the effects of different classification properties using the same underlying reference data without resorting to artificial or manipulated classifications,
- the overwhelming number of produced documents make the legal domain a very attractive field for employing supportive automated solutions and therefore a machine learning scenario in step with actual practice,
- the documents are available in several European languages and are hence very interesting e.g. for the wide field of multi- and cross-lingual text classification (cf. e.g. de Melo and Siersdorfer 2007),
- and, finally, the data is freely accessible (at <http://eur-lex.europa.eu/>)

The database is a very challenging multilabel scenario due to the high number of possible labels (up to 4000), which, for example, exceeds the number of labels in the Reuters databases (cf. Table 2.2) by one order of magnitude. The *EUR-Lex* dataset is publicly available under <http://www.ke.tu-darmstadt.de/resources/eurlex/>.

---

Following the setting in Chapter 4, we evaluated three methods on this task: binary relevance perceptrons, the fast multilabel multiclass perceptron algorithm on the basis of the preceding approach, and the multilabel pairwise perceptrons algorithm. The previous work on using these algorithms for text categorization has shown that the MLPP algorithm outperforms the other two algorithms, while being slightly more expensive in training. However, another key disadvantage of the MLPP algorithm is its need for storing one classifier for each pair of classes. For the EUROVOC categorization, this results in almost 8,000,000 perceptrons, which would make it impossible to solve this task in main memory.

To solve this problem, we introduce and analyze a novel variant that addresses this problem by representing the perceptron in its dual form, i.e. the perceptrons are formulated as a combination of the documents that were used during training instead of explicitly as a linear hyperplane (cf. Section 4.1.4). This reduces the dependency on the number of classes and therefore allows the *Dual MLPP* algorithm to handle the tasks in the *EUR-Lex* database.

We will focus on the label ranking abilities of the different approaches in this study though the dual variant is also adapted in order to use calibration. The reasons are twofold. Firstly, the new approach introduced was explicitly developed in order to handle problems with a large number of labels. As could be already observed in the previous chapters, calibration is susceptible to underestimating the cardinality on these types of problems. Hence, it seems convenient to use external bipartitioning techniques in this particular case, such as it would be necessary for the competing MMP algorithm. However, it is doubtful whether this is actually useful, since, secondly, the *EUR-Lex* database is a typical scenario for label ranking prediction. The high number of labels makes it inappropriate for full-automated classification due to the increased uncertainty in the given mappings as well as in the predictions. A supportive approach where a user select the correct labels from a ranked list of suggestions seems more reasonable in this concrete case. *EUR-Lex* is hence also a suitable representative for Web 2.0 and keyword suggestion tasks (cf. Section 2.9).

## 6.1 The EUR-Lex Repository

The *EUR-Lex/CELEX* (Communitatis Europaeae LEX) Site<sup>49</sup> provides a freely accessible repository for European Union law texts. The documents include the official Journal of the European Union, treaties, international agreements, legislation in force, legislation in preparation, case-law and parliamentary questions. They are available in most of the languages of the EU, and in the HTML and PDF format. A cleaned example document in English is given in Figure 6.1.

---

<sup>49</sup> <http://eur-lex.europa.eu>

## Title and reference

Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs

## Classifications

### EUROVOC descriptor

- data-processing law
- computer piracy
- copyright
- software
- approximation of laws

### Directory code

- 17.20.00.00 Law relating to undertakings / Intellectual property law

### Subject matter

- Internal market
- Industrial and commercial property

## Text

*COUNCIL DIRECTIVE of 14 May 1991 on the legal protection of computer programs (91/250/EEC)*

THE COUNCIL OF THE EUROPEAN COMMUNITIES,

Having regard to the Treaty establishing the European Economic Community and in particular Article 100a thereof,

Having regard to the proposal from the Commission (1),

In cooperation with the European Parliament (2),

Having regard to the opinion of the Economic and Social Committee (3),

Whereas computer programs are at present not clearly protected in all Member States by existing legislation and such protection, where it exists, has different attributes;

Whereas the development of computer programs requires the investment of considerable human, technical and financial resources while computer programs can be copied at a fraction of the cost needed to develop them independently;

Whereas computer programs are playing an increasingly important role in a broad range of industries and computer program technology can accordingly be considered as being of fundamental importance for the Community's industrial development;

...

**Figure 6.1:** Excerpt of a *EUR-Lex* sample document with the CELEX ID 31991L0250. The original document contains more meta-information. We trained our classifiers to predict the EUROVOC descriptors, the directory code and the subject matters based on the text of the document.

**Table 6.1:** Statistics of the *EUR-Lex* datasets. *Label density* indicates the average number of labels per instance  $d$  relative to the total number of classes  $n$ , and *distinct* counts the distinct labelsets found in the dataset  $|\{P_i \mid i = 1 \dots m\}|$ . The collection contains 19,348 documents and the feature space was reduced from over 200,000 to 5,000 features.

dataset name	#classes $n$	avg. labelset size $d$	density $\frac{d}{n}$	distinct
<i>EUR-Lex subject matter</i>	201	2.213	1.101 %	2540
<i>EUR-Lex directory code</i>	410	1.292	0.315 %	1615
<i>EUR-Lex EUROVOC</i>	3956	5.310	0.134 %	16467

### 6.1.1 Retrieval

We retrieved the HTML versions with bibliographic notes recursively from all (non empty) documents in the English version of the *Directory of Community legislation in force*<sup>50</sup> on the 7th of July 2006, in total 19,348 documents. Only documents related to secondary law (in contrast to primary law, the constitutional treaties of the European Union) and international agreements are included in this repository.

The legal form of the included acts are mostly *decisions* (8,917 documents), *regulations* (5,706), *directives* (1,898) and *agreements* (1,597). This version of the dataset differs slightly from that presented in previous works (Loza Mencía and Fürnkranz 2008a,b), which still contained 19,596 documents. Some empty documents that were missed in the previous version and all corrigendums (they contained the same standard text except for one document since they were concerned with translations of the law into other languages than English) have been removed.<sup>51</sup> The updated version can be found under <http://www.ke.tu-darmstadt.de/resources/eurlex/>.

### 6.1.2 Statistics

The bibliographic notes of the documents contain information such as dates of effect and validity, authors, relationships to other documents and classifications. The classifications include the assignment to several *EUROVOC descriptors*, *directory codes* and *subject matters*, hence all classifications are multilabel ones.

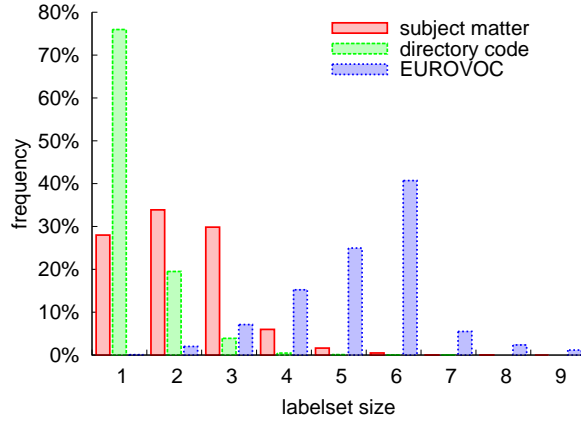
EUROVOC is a multilingual thesaurus providing a controlled vocabulary for European Institutions<sup>52</sup>. Documents in the documentation systems of the EU are indexed using this thesaurus. The EUROVOC thesaurus was already presented in similar tasks

<sup>50</sup> <http://eur-lex.europa.eu/en/legis/index.htm>

<sup>51</sup> Originally, the number of 19,940 documents was retrieved. A range of documents were excluded: 214 contained some error message e.g. that the document was not available in English (189), 316 contained an empty text field, 50 did not contain any relevant category information and the 12 corrigendums were already mentioned.

<sup>52</sup> <http://europa.eu/eurovoc/>





**Figure 6.3:** Distribution of the labelset sizes for the three *EUR-Lex* datasets.

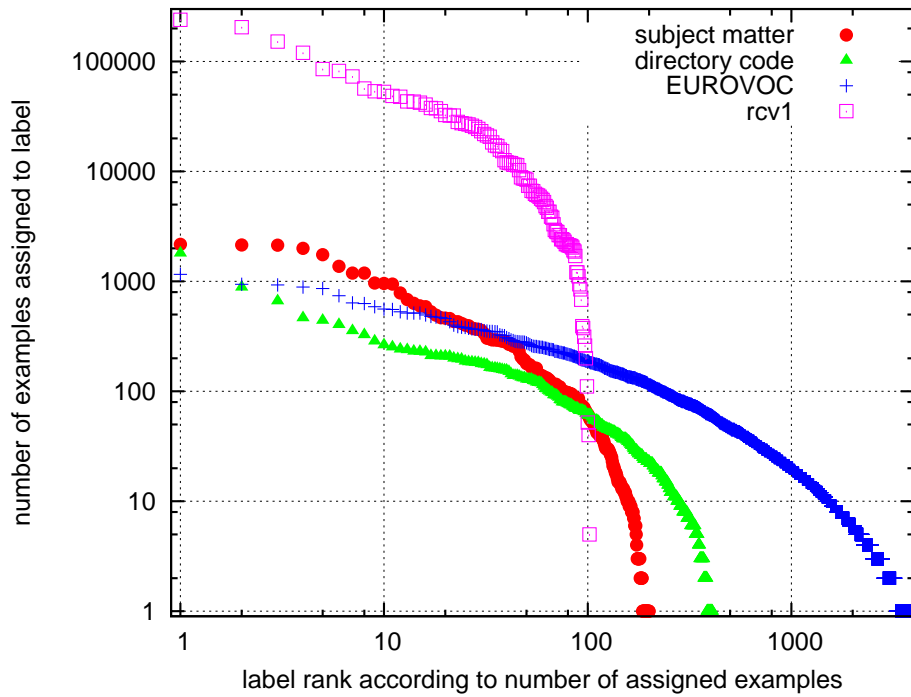
of 2.21 labels per document, and 410 different directory codes, with a labelset size of on average 1.29.<sup>54</sup> Figure 6.3 provides a visualization of the distribution of the labelset sizes. An overview of basic statistics of the different views on the dataset used in this study are given in Table 6.1.

Previous dataset such as the Reuters *rcv1* corpus (cf. Section 4.6) were constructed in order to fulfill certain properties which are beneficial for classification (Rubin et al. 2011). E.g. the categories were carefully selected in order to not exceed too much the number of 100. In addition, it was tried to construct well balanced categories, in particular to include only few categories containing a small number of documents. *EUR-Lex*, in contrast, was not preprocessed in this respect. This is reflected in Figure 6.4, which visualizes the sizes of the labels, i.e. the number of examples associated to the individual labels, for the different *EUR-Lex* subsets as well as for *rcv1*. Apart from the obvious difference in the absolute sizes of the labels, we can observe that the sizes are more evenly distributed for the *EUR-Lex* datasets. The curves are straighter and evenly dense. The only few points below the number of 1000 and 100 examples for *rcv1* in comparison to the *EUR-Lex* datasets demonstrates the distinct distributions and the artificial construction process of *rcv1*.

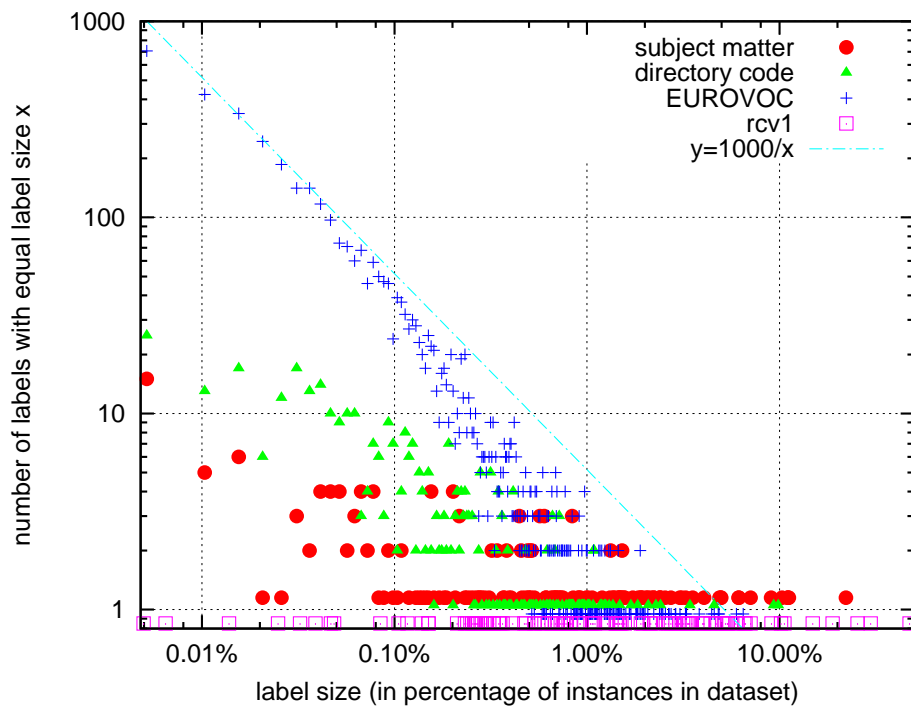
This is confirmed in Figure 6.5, which shows the frequencies of the label sizes. A point  $(x, y)$  in this diagram means that there exist  $y$  labels with the same number of associated examples  $x$ . We can clearly observe that the frequency of the label sizes for the *EUR-Lex* subsets follows a power law. This relationship between an event’s attribute (the size of the labels) and their frequency can be universally observed, e.g. in natural processes, biology, physics, mathematics, social sciences, etc. The power law states that the frequency of an

<sup>54</sup> Note that for the directory codes we only used the assignment to the leaf category as the parent nodes can be deduced from the leaf node assignment. For the document in Figure 6.1 this would mean a set of labels of {17.20} instead of {17, 17.20}. The parent nodes would increase the label set size to 3.75. More details and additional level-wise statistics and subsets can be found on the online site.





**Figure 6.4:** Diagram of the sizes of the labels in number of associated examples for the *EUR-Lex* dataset and Reuters *rcv1* for comparison.



**Figure 6.5:** Distribution of the sizes of the labels. The points of the different datasets on  $y = 1$  were slightly shifted in height for better visibility.

---

event follows as a power of an attribute of that event, i.e. in our case the size. It reflects the intuition that small sizes are measured very often, whereas extreme sizes happen rather infrequently. A popular example is the populations of cities in a country, which usually follow a power law. Another example is Zipf's law, which determines that the frequency of a word in a natural language corpus is inversely proportional to its frequency rank.

As the plot  $y = 1000 \cdot x^{-1}$  demonstrates, particularly the frequencies for *EUROVOC* match the Zipf-distribution quite well. Rubin et al. (2011) point out that this power-law relation is natural for corpora in the real world: these datasets usually contain many rare labels and only a few frequent labels. *rcv1*, in contrast, does not follow the power law distribution, even if the label sizes are bucketed so that one *rcv1* point matches one *EUR-Lex* point (not shown here). *EUR-Lex*, in particular the *EUROVOC* subset, is evidently a good representative of real world data and is hence very well suited for analyzing the behavior of multilabel learning algorithms in realistic scenarios, in particular in comparison to *rcv1*.

### 6.1.3 Preprocessing

Figure 6.1 shows an excerpt of a sample document with all information that has not been used removed. The full document can be viewed at <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31991L0250:EN:NOT>. We extracted the text body from the HTML documents, excluding HTML tags, bibliographic notes or other additional information that could distort the results. The text was tokenized into lower case, stop words were excluded, and the Porter stemmer algorithm was applied. In order to perform cross validation, the instances were randomly distributed into ten folds. The tokens were projected for each fold into the vector space model using the common TF-IDF term weighting (Salton and Buckley 1988, Sebastiani 2002).

In order to reduce the memory requirements, of the approx. 200,000 resulting features we selected the first 5,000 ordered by their document frequency. Though this feature selection method is very simple and efficient and independent from class assignments, it performs comparably to more sophisticated methods using chi-square or information gain computation (Yang and Pedersen 1997). In order to ensure that no information from the test set enters the training phase, the TF-IDF transformation and the feature selection were conducted only on the training sets of the ten cross-validation splits.

## 6.2 Dual Multilabel Pairwise Perceptrons

Perhaps the hardest problem in the context of pairwise multilabel classification is that even if training and testing can be performed efficiently, one still has to store a number of classifiers that is quadratic in the number of potential labels. Even on modern computers with a large memory this problem becomes unsolvable for a high number of classes. For



the *EUROVOC* dataset e.g., the use of MLPP would mean maintaining approximately 8,000,000 perceptrons in memory.

In order to circumvent this obstacle we reformulate the MLPP ensemble of perceptrons in dual form as we did with one single perceptron in Section 4.1.4. Remind that we can formulate the weight vector  $\mathbf{w}$  of a perceptron as a linear combination  $\sum_{i=1}^m \alpha_i \mathbf{x}_i$  of the training examples that were used for updating  $\mathbf{w}$ . In contrast to MLPP, the training examples (the support vectors in SVM terminology) are thus required and have to be kept in memory in addition to the associated weights, since a base perceptron is now represented as  $\mathbf{w}_{u,v} = \sum_{i=1}^m \alpha_{u,v}^i \mathbf{x}_i$ . This makes an additional loop over the training examples inevitable every time a prediction is demanded. But fortunately it is not necessary to recompute all  $\mathbf{x}_i \mathbf{x}$  for each base perceptron since we can reuse them by iterating over the training examples in the outer loop, as can be seen in the following equations:

$$\begin{aligned}
\mathbf{w}_{1,2} \mathbf{x} &= \alpha_{1,2}^1 \mathbf{x}_1 \mathbf{x} + \alpha_{1,2}^2 \mathbf{x}_2 \mathbf{x} + \dots + \alpha_{1,2}^m \mathbf{x}_m \mathbf{x} \\
\mathbf{w}_{1,3} \mathbf{x} &= \alpha_{1,3}^1 \mathbf{x}_1 \mathbf{x} + \alpha_{1,3}^2 \mathbf{x}_2 \mathbf{x} + \dots + \alpha_{1,3}^m \mathbf{x}_m \mathbf{x} \\
&\vdots \\
\mathbf{w}_{1,n} \mathbf{x} &= \alpha_{1,n}^1 \mathbf{x}_1 \mathbf{x} + \alpha_{1,n}^2 \mathbf{x}_2 \mathbf{x} + \dots + \alpha_{1,n}^m \mathbf{x}_m \mathbf{x} \\
\mathbf{w}_{2,3} \mathbf{x} &= \alpha_{2,3}^1 \mathbf{x}_1 \mathbf{x} + \alpha_{2,3}^2 \mathbf{x}_2 \mathbf{x} + \dots + \alpha_{2,3}^m \mathbf{x}_m \mathbf{x} \\
&\vdots
\end{aligned} \tag{6.1}$$

By advancing column by column it is not necessary to repeat the dot products computations, however it is necessary to store the intermediate values, as can also be seen in the pseudocode of the training and prediction phases in Figure 6.6 and 6.7. Note also that the algorithm preserves the property of being incrementally trainable. We denote this variant of training the pairwise perceptrons the *dual multilabel pairwise perceptrons* algorithm (DMLPP).

During the incremental training, the new training instance  $\mathbf{x}_m$  is multiplied in the outer loop with the support vectors (SVs). As in conventional MLPP training (cf. Figure 4.3), only the perceptrons concerned with pairings between relevant  $P$  and irrelevant labelset  $N$  are evaluated. Moreover, it holds that  $\alpha_{u,v}^i = 0$  for  $\{\lambda_u, \lambda_v\} \subseteq P \vee \{\lambda_u, \lambda_v\} \subseteq N$ . Hence, the set of pairings that has to be evaluated is further reduced to  $((P_m \cap P_i) \times (N_m \cap N_i)) \cup ((P_m \cap N_i) \times (P_m \cap N_i))$ . Added to the possibility that the current support vector was not implicated as training instance in any of the base perceptrons, it may occur that it is not necessary to compute the dot product. Following this procedure, it is ensured in Line 5 in Figure 6.6 that  $s_{u,v}$  results in  $\mathbf{w}_{u,v} \mathbf{x}_m = \sum_{i=1}^{m-1} \alpha_{u,v}^i \mathbf{x}_i \mathbf{x}_m$  at the end of both loops. In the following lines we repeat the loop over the pairings and evaluate if  $\mathbf{w}_{u,v}$  mistakenly would have predicted label  $\lambda_v \in N_m$  as positive.<sup>55</sup> In that case, we simply determine the current training example as positive support vector for  $\mathbf{w}_{u,v}$ . Thus, setting  $\alpha_{u,v}^m \leftarrow 1$

<sup>55</sup> In our implementation we perform an update randomly in half of the cases if  $s_{u,v} = 0$ . This corresponds to a random weight vector initialization in MLPP and random assignment in the case of a tie (cf. Footnote 34).

---

**Require:** New training example pair  $(\mathbf{x}_m, P_m)$ ,  
training examples  $\langle (\mathbf{x}_1, P_1), \dots, (\mathbf{x}_{m-1}, P_{m-1}) \rangle$ ,  
weights  $\{\alpha_{u,v}^i \mid \lambda_u, \lambda_v \in \mathcal{L}, 1 \leq i < m\}$

```

1: for each  $\mathbf{x}_i = \mathbf{x}_1 \dots \mathbf{x}_{m-1}$  do                                ▷ iterate over previous training examples
2:   for each  $(\lambda_u, \lambda_v) \in P_m \times N_m$  do                        ▷  $\mathbf{x}_m$  only relevant for training these pairs
3:     if  $\alpha_{u,v}^i \neq 0$  then                                          ▷ also ensures that  $(\lambda_u, \lambda_v) \in P_i \times N \vee (\lambda_v, \lambda_u) \in P_i \times N$ 
4:       if  $p_i$  still undefined then  $p_i \leftarrow \mathbf{x}_i \cdot \mathbf{x}_m$       ▷ compute  $p_i$  only if  $\exists u, v. \alpha_{u,v} \neq 0$ 
5:        $s_{u,v} \leftarrow s_{u,v} + \alpha_{u,v}^i \cdot p_i$                     ▷ note that  $s_{u,v} = -s_{v,u}$ 
6:   for each  $(\lambda_u, \lambda_v) \in P_m \times N_m$  do                        ▷ update only concerned perceptrons
7:     if  $s_{u,v} < 0$  then                                            ▷ and only if they misspredicted
8:        $\alpha_{u,v}^m \leftarrow 1$                                        ▷ note that  $\alpha_{u,v} = -\alpha_{v,u}$ 
9: return  $\{\alpha_{u,v}^m \mid (\lambda_u, \lambda_v) \in P \times N\}$                 ▷ return new weights

```

---

**Figure 6.6:** Pseudocode of the incremental training method of the DMLPP algorithm.

corresponds to  $\mathbf{w}_{u,v}^m = \mathbf{w}_{u,v}^{m-1} + \mathbf{x}_m$  in primer MLPP. Indeed, negative support vectors are also obtained due to the symmetry  $\alpha_{u,v} = -\alpha_{v,u}$ .

The prediction phase reproduced in Figure 6.7 follows a similar setup. Firstly, we iterate over the potential support vectors, which corresponds to the columns in Eq. 6.1. Secondly, we identify which perceptrons  $\mathbf{w}_{u,v}$  (this time, all pairings  $(\lambda_u, \lambda_v) \in P_i \times N_i$  have to be evaluated) actually make use of the support vector  $\mathbf{x}_i$  and we correspondingly update the accumulated scores  $\mathbf{w}_{u,v} \cdot \mathbf{x}$ . Lastly, we iterate over all scores  $s_{u,v}$  as it is done with  $h_{u,v}(\mathbf{x})$  in conventional pairwise classification (cf. Section 3.4.3) and hence obtain the vote vector.

### 6.2.1 Calibration

There exist two ways of adapting the calibration approach described in Section 3.4.5 for DMLPP: processing the additional subproblems internally or externally.

The first version just includes the artificial label  $\lambda_0$  to the set of possible labels  $\mathcal{L}$  and extends the iteration in line 3 and 6 of the training algorithm in Figure 6.6 by adding the combinations between this label and all the others  $\{\lambda_0\} \times \mathcal{L} \setminus \{\lambda_0\}$  to the combinations between positive and negative classes  $P \times N$ . However, we believe that this approach could decrease the advantage that DMLPP obtains through the *sparseness* of the support vectors, which is due to the pairwise decomposition. By sparseness we mean the relation between the full training set and the actually used support vectors  $\{\mathbf{x}_i \mid \exists \alpha_{u,v}^i \neq 0\}$ . Pairwise subproblems considering the artificial class are larger and hence more difficult to solve. This normally results in more training examples that have to be used to update the model, and hence more SVs. The higher density (the lower sparseness) of SVs directly increases the amount of dot products needed for processing both training and test examples.

Therefore, the second version considered simply trains an external (non-dual) binary relevance classifier (as described in Section 3.1) in parallel. During classification, the

---

**Require:** example  $\mathbf{x}$  for classification,  
training examples  $((\mathbf{x}_1, P_1), \dots, (\mathbf{x}_m, P_m))$ ,  
weights  $\{\alpha_{u,v}^i \mid \lambda_u, \lambda_v \in \mathcal{L}, 1 \leq i \leq m\}$

- 1: **for each**  $\mathbf{x}_i = \mathbf{x}_1 \dots \mathbf{x}_{m-1}$  **do** ▷ iterate over training examples
- 2:    $p \leftarrow \mathbf{x}_i \cdot \mathbf{x}$
- 3:   **for each**  $(\lambda_u, \lambda_v) \in P_i \times N_i$  **do** ▷  $\mathbf{x}_i$  was only be part of training these pairs
- 4:     **if**  $\alpha_{u,v}^i \neq 0$  **then** ▷ consider only if  $\mathbf{x}$  is actually part of  $\mathbf{w}_{u,v}$
- 5:       **if**  $p_i$  undefined **then**  $p_i \leftarrow \mathbf{x}_i \cdot \mathbf{x}_m$  ▷  $p_i$  only needed if  $\mathbf{x}_i$  was SV for a  $\mathbf{w}_{u,v}$
- 6:        $s_{u,v} \leftarrow s_{u,v} + \alpha_{u,v}^i \cdot p$  ▷ add intermediate score to  $\mathbf{w}_{u,v}$
- 7: **for each**  $(\lambda_u, \lambda_v) \in \mathcal{L} \times \mathcal{L}, u \neq v$  **do**
- 8:    $v_u \leftarrow \begin{cases} v_u + 1 & s_{u,v} > 0 \\ v_u + \frac{1}{2} & s_{u,v} = 0 \end{cases}$  ▷ add up a (half) vote for winning (tied) class  $\lambda_u$
- 9: **return** voting  $\mathbf{v} = (v_1, \dots, v_{|\mathcal{L}|})$  ▷ return vote vector

---

**Figure 6.7:** Pseudocode of the prediction phase of the DMLPP algorithm.

predictions of the base perceptrons of the BR classifier are incorporated in the voting process. We will denote this algorithm as *DCMLPP*.

### 6.2.2 Discussion and Further Extensions

Remind that the described decomposition and aggregation in DMLPP results in an absolutely mathematical equivalence to MLPP. Let  $h_{\mathcal{T}_{rain}}^P$  be the primal version of MLPP trained on a given arbitrary training set  $\mathcal{T}_{rain}$ , let  $h_{\mathcal{T}_{rain}}^D$  be correspondingly the dual variant DMLPP, and let us assume, without loss of generality, an equivalent initialization  $\mathbf{w}_{u,v}^0 = \mathbf{x}_0 := (1, \dots, 1)$  for MLPP and  $\alpha_{u,v}^0 = 1$  for DMLPP,  $1 \leq u < v \leq n$ , and equivalent behavior at ties. Then it holds that

$$\forall \mathbf{x} \in \mathcal{X}. \quad h_{\mathcal{T}_{rain}}^P(\mathbf{x}) = h_{\mathcal{T}_{rain}}^D(\mathbf{x}) \quad (6.2)$$

This follows directly from tracking the computations for the decision hyperplanes  $\mathbf{w}_{u,v}^m$ ,  $m = |\mathcal{T}_{rain}|$ . In both cases we obtain

$$\mathbf{w}_{u,v}^m = \sum_{k=0}^m \alpha_{u,v}^k \mathbf{x}_k \quad \alpha_{u,v}^0 = 1 \quad \alpha_{u,v}^k = \begin{cases} 0 & \{\lambda_u, \lambda_v\} \subseteq P_k \vee \{\lambda_u, \lambda_v\} \subseteq N_k \\ 1 & \left( \sum_{i=0}^k \alpha_{u,v}^i \mathbf{x}_i \right) \cdot \mathbf{x}_k < 0 \wedge \lambda_u \in P_k \\ -1 & \left( \sum_{i=0}^k \alpha_{u,v}^i \mathbf{x}_i \right) \cdot \mathbf{x}_k > 0 \wedge \lambda_u \in N_k \\ 0 & \text{otherwise} \end{cases} \quad (6.3)$$

already compactly represented in Eq. 6.1.<sup>56</sup> We abstain from giving a tedious complete formal proof.

<sup>56</sup> For the sake of clarity, and without loss of generality, we omit the case of equality and assume that the continuous value of the sum of the dot products is never zero. See also Footnote 34.

Note also that the pseudocode needs to be slightly adapted when the DMLPP algorithm is trained in more than one epoch, i.e. the training set is presented to the learning algorithm more than once (cf. Section 4.1.2). It is sufficient to modify the assignment in line 8 in Figure 6.6 to an additive update  $\alpha_{u,v}^i = \alpha_{u,v}^i + 1$  for a revisited example  $\mathbf{x}_{T \cdot m+i} = \mathbf{x}_i$ . This setting is particularly interesting for the dual variant since, when the training set is not too big, memorizing the inner products can boost the subsequent epochs in a substantial way, making the algorithm interesting even if the number of classes is small.

Hybrid variants of the primer and dual MLPPs could further reduce the computational complexity. The idea is to use a different formulation in training than in the prediction phase depending on the specific memory and runtime requirements of the classification task. In order e.g. to combine the advantage of MLPP during training and DMLPP during predicting on the *subject matter* subproblem (cf. Section 6.4), we could train the classifier as in the MLPP (with the difference of iterating over the base classifiers first and then over the instances instead of reversely so that only one perceptron has to remain in memory) and then convert it by means of the collected information during training the perceptrons to the dual representation.

In addition to the savings in memory and run-time, analyzed in detail in Section 6.3, the dual representation easily allows for using the kernel trick, i.e. to replace the dot product by a kernel function, in order to be able to solve originally not linearly separable problems. However, this is not necessary in the case of *EUR-Lex* since text problems are in general linearly separable (cf. Section 4.1.6).

It follows from the mathematical equivalence that it should be possible to use more advanced versions of the perceptron algorithm. Most of the approaches cited in Footnote 36 concentrate on the update process, so that only line 8 has to be adapted. Others focus on the minimization or even budgeting of the number of support vectors used (e.g. Dekel et al. 2005, Orabona et al. 2009), a measure from which DMLPP would directly benefit. Moreover and following the hybrid idea, the  $\alpha$  weights in DMLPP could be obtained by sequentially learning the pairwise classifiers using SVMs. The sparseness, and hence the efficiency, could be further improved by relaxing the costs of using support vectors already deployed by a preceding pairwise linear classifier (the second sum in Eq. 4.6). This also would apply to conventional perceptron algorithms. A further investigation in this direction deserves increased attention.

An application of QVoting in the conventional way as in Chapter 5 is unrewarding, since the scores are already computed when the voting process starts. But the implementation in the process of accumulation of the scores, intelligently selecting promising or omitting unpromising dot products, seems a practicable approach which could be investigated in the future.

### 6.3 Computational Complexity

We recall the notation used for the complexity analysis:  $n$  denotes the number of possible classes,  $d$  the average number of relevant classes per instance in the training set,  $a$  the number of attributes and  $a'$  the average number of attributes not zero (size of the sparse

**Table 6.2:** Computational complexity of the perceptron based algorithms given as upper bounds of number of addition and multiplication operations, for each instance, and in terms of the number of classes  $n$ , the average cardinality  $d$ , the  $m$  training examples, the  $a$  attributes and the average number of attributes  $a'$  not zero.

	training time	prediction time	memory requirement
MMP/ BR	$\mathcal{O}(na')$	$\mathcal{O}(na')$	$\mathcal{O}(na)$
MLPP	$\mathcal{O}(dna')$	$\mathcal{O}(n^2a')$	$\mathcal{O}(n^2a)$
QCMLPP	$\mathcal{O}(dna')$	$\sim na' + dn \log(n) a'$	$\mathcal{O}(n^2a)$
DMLPP	$\mathcal{O}(m(dn + a'))$	$\mathcal{O}(m(dn + a'))$	$\mathcal{O}(m(dn + a') + n^2)$

representation of an instance), and  $m$  denotes the size of the training set. As in previous analysis, we do not estimate  $d$  as  $\mathcal{O}(n)$  and distinguish between  $a$  and  $a'$ .

We will indicate the runtime complexity in terms of real value additions and multiplications ignoring operations that have to be performed by all algorithms such as sorting or internal real value operations. Note that while the scalar multiplication with a linear model  $\mathbf{w}\mathbf{x}$  costs around  $2a'$  real value operations, the products between examples  $\mathbf{x}_i\mathbf{x}_j$  in DMLPP usually require much less operations since the probability is high that at least one vector component is zero. The relation is e.g. 290 vs. 56 operations in the *EUR-Lex* dataset. However, we estimate both types of operations in the same manner as  $\mathcal{O}(a')$ .

We will present the complexities per instance since all algorithms are incrementally trainable. We will also include the estimations for the QVoting variant of CMLPP.

### 6.3.1 Memory Requirements

BR and MMP follow an one model per class approach, so they have to keep the same amount of perceptrons in memory, leading to  $\mathcal{O}(n \cdot a)$  memory space. In contrast, the non-dual pairwise approaches require one perceptron for each of the  $\frac{n(n-1)}{2}$  pairs of classes, hence we need  $\mathcal{O}(n^2a)$  memory. In addition, the calibrated versions require an overhead of  $n$  perceptrons for the comparisons with the artificial label. The same values apply for the QVoting variants.

In the worst case when all training examples are needed as support vectors, the DMLPP algorithms keeps the whole training set in memory, and additionally requires for each training example  $\mathbf{x}$  access to the weights of all class pairs  $P \times N$ . Furthermore, it has to intermediately store the resulting scores for each base perceptron during prediction, hence the complexity is  $\mathcal{O}(mdn + ma' + n^2) = \mathcal{O}(m(dn + a') + n^2)$ . Note that usually  $a' \ll a$ , for the *EUR-Lex* dataset in particular  $290 \ll 5000$ . In fact, we chose the 5000 most frequent features from a set of more than 200,000.

We can see that (QC)MLPP is applicable especially if the number of classes is low and the number of examples high, whereas DMLPP is suitable when the number of classes is high, however it does not handle huge training sets very well.

---

### 6.3.2 Training

For processing one training example, we recall that  $n$  dot products have to be computed by BR and MMP, plus at most the same amount if there were prediction errors. The non-dual MLPPs require  $|P|(n - |P|)$  dot products, one for each associated perceptron. Assuming that a dot product computation costs  $\mathcal{O}(a')$ , we obtain a complexity of  $\mathcal{O}(dna')$  per training example.

Similarly, the DMLPP spends  $m$  dot product computations in the worst case. Note that this is a very pessimistic estimation, since in practice the set of support vectors may be much smaller (sparseness). In addition, it is probable that the pairings from the training instance  $P \times N$  do not coincide with those of the support vector  $\mathbf{x}_i$ , i.e. that  $(\lambda_u, \lambda_v) \in ((P_m \cap P_i) \times (N_m \cap N_i)) \cup ((P_m \cap N_i) \times (P_m \cap N_i))$  is empty, as can be seen in the empirical evaluation (cf. Section 6.4). Even though, the summation of the scores is asymptotically bounded by  $\mathcal{O}(dn)$  per support vector, leading to  $\mathcal{O}(m(dn + a'))$  operations. It is obvious that MLPP has a clear advantage over DMLPP in terms of training time, unless  $n$  is of the order of magnitude of  $m$  or the model is trained over several epochs, as already outlined in the previous Section 6.2.

### 6.3.3 Predicting

During prediction the MLPP evaluates all perceptrons, leading to  $\mathcal{O}(n^2 a')$  computations. The dual variant again iterates over all support vectors and associated weights, hence the complexity is  $\mathcal{O}(m(dn + a'))$ . At this phase DMLPP benefits from the linear dependence of the number of classes in contrast to the quadratic relationship of the MLPP. Roughly speaking, the breaking point when DMLPP is faster in prediction is approximately when the square of the number of classes is clearly greater than the number of training documents. Of course this does not hold in the same degree for the comparison with the QVoting improvement. QCMLPP requires empirically at most  $na' + dna' \log n$  computations (cf. Section 5.2). For BR and MMP with its  $\mathcal{O}(na')$  computations, DMLPP becomes more efficient roughly when  $m$  is below  $a'$  and  $n$ , i.e. if only sparse training data is available.

We can find a similar trade-off between primal and dual MLPP for the memory requirements with the difference that the factor between sparse and total number of attributes becomes more important, leading earlier to the breaking point when the sparseness is high.

A compilation of the analysis can be found in Table 6.2, together with the complexities of MMP and BR. For the complexities of the calibrated variants of MLPP and DMLPP we can simply add the corresponding complexity of BR, at least if we consider the externally calibrated variant of DCMLPP.

In summary, it can be stated that the dual form of the MLPP balances the relationship between training and prediction time by increasing training and decreasing prediction costs, and especially benefits from a decreased prediction time and memory savings

---

when the number of classes is large. Thus, this technique addresses the main obstacle to applying the pairwise approach to problems with a large number of labels.

## 6.4 Evaluation

The following sections presents the setup of our experimental evaluation and analyzes the results regarding predictive quality and computational costs.

### 6.4.1 Experimental Setup

For the MMP algorithm we used the  $\text{IsErr}$  loss function and the uniform penalty function. All the perceptrons of the different algorithms were initialized with random values, except for the dual variant for which updating the base classifier for the first time was randomized. This simulates the behavior of MLPP, but obviously results in slightly different results though using the same random seed.

We performed also tests with a multilabel variant of the multinomial Naive Bayes (MLNB) algorithm in order to provide a baseline (cf. Section 4.6.1). Another baseline is depicted by FC (frequency classifier) which returns always the same ranking of classes according to the class frequency in the training set.

DMLPP results are omitted since they differ only slightly from those of DCMLPP due to the possible additional (one) vote won against the artificial label. In the same way, we omit the results of MLPP since they differ only marginally due to a different random initialization. Note however that MLPP cannot be applied to the *EUROVOC* dataset due to the high memory requirements, which was the reason for developing the dual version.

The number of epochs indicates the number of times that the online-learning algorithms were able to see the training instances. No results are reported for the performance of DCMLPP on *EUROVOC* for more than two epochs due to time restrictions. Note also that the results differ slightly from those of previous experiments of [Loza Mencía and Fürnkranz \(2008a,b\)](#) due to the modifications to the dataset presented in Section 6.1. All results are averaged from the results of the ten fold cross validation (cf. Section 6.1).

### 6.4.2 Ranking Quality

The results for the four algorithms and the three different classifications of *EUR-Lex* are presented in Table 6.3 in terms of the ranking losses  $\text{IsErr}$ ,  $\text{OneErr}$ ,  $\text{RankLoss}$ ,  $\text{AvgP}$ ,  $\text{F1}_{|P|}$  (in percentage for better readability) and  $\text{Margin}$  (cf. Section 2.7.4).

The first appreciable characteristic is that DCMLPP dominates all other algorithms on all three views of the *EUR-Lex* data, regardless of the number of epochs or losses. Often DCMLPP achieves better results than the other algorithms for more epochs. Especially on the losses that directly evaluate the ranking performance the improvement is quite pronounced and the results are already unreachable after the first epoch.



**Table 6.3:** Average ranking losses for the three views on the data and for the different algorithms. For  $IS_{ERR}$ ,  $ONE_{ERR}$ ,  $RANK_{LOSS}$  and  $MARGIN$  low values are good, for  $AVG_P$  and  $F1_{|p|}$  the higher the better. Bold values indicate the best value with respect to the number of epochs.

	1 epoch					2 epochs			5 epochs			10 epochs			
	FC	MLNB	BR	MMP	DCMLP	BR	MMP	DCMLP	BR	MMP	DCMLP	BR	MMP	DCMLP	
subject matter	IS <sub>ERR</sub>	99.58	99.47	65.99	55.70	51.38	58.78	51.96	44.07	53.42	42.77	38.23	50.19	40.22	36.34
	ONE <sub>ERR</sub>	77.83	98.68	35.71	30.58	22.78	27.13	27.09	17.29	22.69	18.38	13.49	20.64	15.97	12.55
	RANK <sub>LOSS</sub>	12.89	8.885	17.38	2.303	1.064	13.89	2.520	0.911	11.58	2.091	0.796	9.752	1.850	0.762
	MARGIN	40.16	25.04	62.31	10.11	4.316	52.28	11.22	3.757	44.77	9.366	3.337	38.45	8.177	3.214
	AVGP	22.57	11.91	59.33	74.01	78.68	66.07	76.95	82.73	70.69	82.10	85.64	73.30	83.75	86.52
F1 <sub> p </sub>	19.61	1.88	54.79	64.61	70.07	60.79	68.54	74.56	65.64	74.33	78.18	68.45	76.15	79.34	
directory code	IS <sub>ERR</sub>	91.51	99.34	52.80	47.68	36.55	46.26	40.01	32.38	40.76	33.28	29.22	37.55	31.39	28.30
	ONE <sub>ERR</sub>	90.13	99.04	44.40	40.85	28.22	37.38	32.99	24.42	31.48	25.79	21.41	28.1	23.9	20.65
	RANK <sub>LOSS</sub>	14.17	7.446	19.40	2.383	0.972	15.09	2.058	0.863	11.69	1.874	0.824	9.876	1.529	0.815
	MARGIN	68.33	34.44	96.43	14.18	5.626	77.32	12.18	5.045	61.48	10.95	4.831	52.94	8.947	4.785
	AVGP	18.98	6.714	57.10	68.70	77.89	63.68	74.90	80.87	68.75	79.84	82.87	71.61	81.30	83.38
F1 <sub> p </sub>	8.47	0.93	49.37	55.08	67.19	56.37	63.29	71.27	61.83	70.00	74.19	64.83	71.86	75.04	
EUROVOC	IS <sub>ERR</sub>	99.82	99.82	99.25	99.14	98.20	98.70	98.00	96.75	97.46	96.14		97.06	95.13	
	ONE <sub>ERR</sub>	93.52	99.58	53.11	78.98	34.76	44.93	56.88	28.01	36.69	39.46		33.84	34.99	
	RANK <sub>LOSS</sub>	12.97	22.34	39.78	3.669	2.692	35.25	4.091	2.398	30.93	4.573		28.59	4.509	
	MARGIN	1357.10	1623.72	3218.12	562.81	426.28	3040.01	670.65	387.51	2846.47	757.01		2716.63	740.12	
	AVGP	5.504	1.060	25.55	27.04	46.79	30.71	38.42	52.72	35.95	47.65		38.31	50.71	
F1 <sub> p </sub>	5.646	0.427	28.67	24.76	43.07	33.64	35.16	48.04	38.61	44.24		40.81	47.07		



---

In addition to the fact that the DMLPP outperforms the remaining algorithms, it is still interesting to compare the performances of MMP and BR as they have still the advantage of reduced computational costs and memory requirements in comparison to the (dual) pairwise approach and could therefore be more applicable for very complex datasets such as *EUROVOC*, which is certainly hard to tackle for DMLPP (cf. Section 6.4.4).

For the *subject matter* and *directory code*, the results clearly show that the MMP algorithm outperforms the simple one-against-all approach. Especially on the losses that directly evaluate the ranking performance the improvement is quite pronounced. The smallest difference can be observed in terms of  $\text{ONEERR}$ , which evaluates the top class accuracy.

The performance on the *EUROVOC* descriptor dataset confirms the previous results. The differences in  $\text{RANKLOSS}$  and  $\text{MARGIN}$  are very pronounced. In contrast, in terms of  $\text{ONEERR}$  the MMP algorithm is worse than one-against-all, even after ten epochs. It seems that with an increasing amount of classes, the MMP algorithm has more difficulties to push the relevant classes to the top such that the margin is big enough to leave all irrelevant classes below, although the algorithm in general clearly gives the relevant classes a higher score than the one-against-all approach. An explanation could be the dependence between the perceptrons of the MMP. This leads to a natural normalization of the scalar product, while there is no such restriction when trained independently as done in the binary relevance algorithm. As a consequence there could be some perceptrons that produce high maximum scores and thereby often arrive at top positions at the overall ranking. Furthermore, MMP's accuracy on  $\text{RANKLOSS}$  and  $\text{MARGIN}$  seems to suffer from the increased number of classes, since the loss increases from the first to the fifth epoch, and still in the tenth epoch the value is higher than after only one epoch. Perhaps it is indicated to use a different loss for MMP to optimize for problems with higher amount of classes, where  $\text{ISERR}$  is inevitably high (cf. Section 4.3). The price to pay for the good  $\text{ONEERR}$  of BR is a decreased quality of the produced rankings, as the results for  $\text{RANKLOSS}$  and  $\text{MARGIN}$  are even beaten by Naive Bayes, which is by far the worst algorithm for the other losses.

It is interesting to note in this context that the frequency classifier often achieves a better performance than Naive Bayes and even BR, especially with increasing number of classes as with *EUROVOC*.

The fact that in only approximately 5% of the cases a perfect classification is achieved and in only approx. 65% the top class is correctly predicted in *EUROVOC* (MMP) should not lead to an underestimation of the performance of these algorithms. Considering that with almost 4000 possible classes and only 5.3 classes per example the probability of randomly choosing a correct class is less than one percent, namely 0.13%, the performance is indeed substantial.

### 6.4.3 Bipartition Prediction Quality

Table 6.4 shows the several results for predicting a set of labels for each instance rather than a ranking of labels. Obviously, only results for BR and the calibrated version of

**Table 6.4:** Average bipartitioning losses for the three views on the data and for the label set predicting BR and DCMLPP. For HAMLoss low values are good, for the remaining measures high values near 100% are good. Bold values indicate the best value with respect to the number of epochs.

		1 epoch		2 epochs		5 epochs		10 epochs	
		BR	DCMLPP	BR	DCMLPP	BR	DCMLPP	BR	DCMLPP
<i>subject matter</i>	HAMLoss	1.196	<b>0.715</b>	1.004	<b>0.641</b>	0.823	<b>0.574</b>	0.757	<b>0.540</b>
	F1	54.39	<b>62.43</b>	60.13	<b>68.81</b>	65.73	<b>72.66</b>	68.32	<b>74.47</b>
	REC	<b>64.64</b>	54.02	<b>68.66</b>	64.26	<b>71.62</b>	69.25	<b>74.11</b>	71.56
	PREC	47.03	<b>74.08</b>	53.55	<b>74.09</b>	60.74	<b>76.43</b>	63.39	<b>77.63</b>
<i>directory code</i>	HAMLoss	0.416	<b>0.231</b>	0.355	<b>0.198</b>	0.289	<b>0.179</b>	0.265	<b>0.169</b>
	F1	46.81	<b>49.37</b>	53.28	<b>62.95</b>	59.74	<b>67.75</b>	62.58	<b>69.64</b>
	REC	<b>58.31</b>	36.05	<b>64.51</b>	53.55	<b>68.36</b>	59.87	<b>70.54</b>	61.89
	PREC	39.13	<b>78.56</b>	45.41	<b>76.38</b>	53.07	<b>78.04</b>	56.26	<b>79.61</b>
<i>EUROVOC</i>	HAMLoss	0.267	<b>0.125</b>	0.238	<b>0.117</b>	0.208		0.199	
	F1	<b>26.95</b>	18.20	31.56	<b>36.11</b>	36.42		38.57	
	REC	<b>37.03</b>	10.45	<b>41.30</b>	24.89	44.84		46.93	
	PREC	21.19	<b>71.62</b>	25.54	<b>65.82</b>	30.67		32.74	

DMLPP can be shown since MMP only produces a ranking. The first remarkable point is that DCMLPP outperforms BR in all direct comparisons for the overall measures HAMLoss and F1 and also PREC. But interestingly, BR always achieves a higher REC than DCMLPP. This is due to the fact that the calibration tends to underestimate the number of returned labels for each instance, especially for a high label density and when the base classifiers are not yet that accurate such as for low numbers of epochs (cf. Section 4.6.9 and solutions therein).

The average labelset size that is produced by DCMLPP demonstrates this: for *subject matter* it increases from 1.65 to 2.04 (BR from 3.05 to 2.59), for *directory code* from 0.59 to 1.0 (BR: 1.93 to 1.62) and for *EUROVOC* it increases from small 0.77 to 2.01 after the second epoch (BR from 9.28 to 7.61 in the tenth epoch). BR begins with an overestimation, reducing the predicted size subsequently.

In order to allow a comparison independent of different tendencies of the different thresholding techniques, we also show the quasi break even point  $F1_{|p|}$  in Table 6.3. This also allows to compare to MMP, which is beaten by DCMLPP but performs better than BR.

#### 6.4.4 Computational Costs

In order to allow a comparison independent from external factors such as logging activities and the run-time environment, we ignored minor operations that have to be performed by all algorithms, such as sorting or internal operations. An overview over the amount of real value addition and multiplication computations is given in Table 6.5 (averaged over the cross validation splits, trained for one epoch), together with the CPU-

**Table 6.5:** Computational costs in CPU-time and millions of real value operations (M op.) on EUR-Lex. The values in parenthesis are only estimated.

	<i>subject matter</i>		<i>directory code</i>		<i>EUROVOC</i>	
	training	testing	training	testing	training	testing
BR	29.96 s 1,680 M op.	7.09 s 184 M op.	50.67 s 3,420 M op.	9.62 s 378 M op.	368.02 s 33,074 M op.	53.34 s 3,662 M op.
MMP	31.95 s 1,807 M op.	6.89 s 184 M op.	53.38 s 3,615 M op.	9.46 s 378 M op.	479.14 s 40,547 M op.	52.90 s 3,662 M op.
DMLPP	372.14 s 6,035 M op.	151.98 s 4,471 M op.	383.40 s 3,047 M op.	187.65 s 5,246 M op.	13,058.01 s 17,647 M op.	6,780.51 s 123,422 M op.
MLPP	69.50 s 3,886 M op.	164.04 s 18,427 M op.	120.70 s 4,735 M op.	643.34 s 77,629 M op.	– (175 G op.)	– (7 · 10 <sup>12</sup> op.)
QCMLPP	86.83 s 5,566 M op.	35.21 761 M op.	159.39 s 8,155 M op.	78.3 s 1,053 M op.	– (209 G op.)	– (74 G op.)

times on an AMD Dual Core Opteron 2000 MHz as additional reference information. We report only results of DMLPP, since DCMLPP’s operations and seconds can easily be derived or estimated by adding those of BR. Furthermore, we include the results for the non-dual MLPP and QCMLPP2, however no values have been received for the *EUROVOC* problem due to the memory space problem discussed at the end of this section.

#### 6.4.4.1 Training and Prediction Costs

We can observe a clear advantage of the non-pairwise approaches on the *subject matter* data especially for the prediction phase, however the training costs are in the same order of magnitude. Between MLPP and DMLPP we can see an antisymmetric behavior: while MLPP requires only almost half of the amount of the DMLPP operations for training, DMLPP reduces the amount of prediction operations by a factor of more than 4. Nevertheless this value is beaten by far by the QVoting variants, repeating the previous observations on the number of base classifier evaluations. As already shown in Section 5.4, the QVoting strategy is competitive to the label-focused approaches in terms of testing time.

For the *directory code* the rate for MMP and BR more than doubles in correspondence with the increase in number of classes. Additionally the MLPP testing time substantially increases due to the quadratic dependency, while DMLPP profits from the decrease in the average number of classes per instance. It even causes less computations in the training phase than MMP/BR. Again QCMLPP is faster in testing than DMLPP, but the distance is quite smaller. For training DMLPP is clearly faster. Note that the calibrating variants need the sum of BR’s and MLPP’s computations for training. The reason for the low number of computations for DMLPP is not only the reduced maximum amount of weights per SV (cf. Section 6.4.4.2), but particularly the decreased probability that a training example is relevant for a new training example (and consequently that dot products and scores have

---

to be computed) since it is less probable that both class assignments match, i.e. that both examples have the same pair of positive and negative classes.

The classifier for *subject matter* has on average 20 weights set per support vector out of 440 ( $= d(n - d)$ ) in the worst case (a ratio of 4.45%), and on average 4.97% of them are required when a new training example arrives. For the *directory code* with a smaller fraction  $d/n$  35.0 weights are stored (6.66%), of which only 1.10% are used when updating. This also explains the relatively small number of operations for training on *EUROVOC*, since from the 1,781 weights per SV (8.45%), only 0.55% are relevant to a new training instance. In this context, regarding the disturbing ratio between real value operations and CPU-time for training DMLPP on *EUROVOC*, we believe that this is caused by a suboptimal storage structure and processing of the weights (and perhaps also a more exhaustive statistics logging for the sake of analysis than on the remaining approaches) and we are therefore confident that it is possible to reduce the distance to MMP in terms of actual consumed CPU-time by improving the program code. Memory swapping may also have influenced the measurement.

However, DMLPP cannot benefit in the same manner from this point during the prediction phase on the *EUROVOC* as on the *directory code* subset. Nevertheless it is still more efficient during training than the one-per-class variants.

Moreover, DMLPP enables to apply pairwise learning to this extremely large subset of classes. Indeed, no results could be retrieved for the non-dual pairwise variants on the *EUR-Lex* dataset due to the high memory requirements (cf. Section 6.4.4.2), but we can try to estimate the expected number of computations. Based on the estimation that MLPP requires  $d$  times computations than BR, we could expect around 175,000 M op. for training on *EUROVOC*. (Q)CMLPP would require additionally the computations of BR. For testing, we estimate the number of base classifier evaluations with  $n + dn \log n$  (cf. Section 5.2) which provides reliable results. Under this assumption and using an average of 0.926 M op. per base classifier evaluation (for the whole test set), we obtain approx. 74,000 M op. for QCMLPP (whereas complete voting would spend more than absurd  $7 \cdot 10^{12}$  op). This would mean again an advantage over DMLPP, but a relatively small one compared to the previous datasets. However for training, DMLPP's costs are almost ten times smaller than the estimated MLPP costs. Note also that such experimental settings are (currently) not possible due to the exponential memory requirements of the non-dual pairwise approaches (see Section 6.4.4.2).

Note that MMP and BR compute the same amount of dot products, the computational costs only differ in the number of vector additions, i.e. perceptron updates. A deeper analysis of the contrary behavior of both algorithms when the number of classes increases can be found in the work of [Loza Mencía and Fürnkranz \(2007a\)](#).

**Table 6.6:** Memory requirements of the different classifiers for the *EUR-Lex* datasets. The values in parenthesis are only estimated.

dataset	BR/MMP	DMLPP	DCMLPP	MLPP
<i>subject matter</i>	153 MB	199 MB	210 MB	541 MB
<i>directory code</i>	167 MB	210 MB	229 MB	1,818 MB
<i>EUROVOC</i>	1,145 MB	1,242 MB	1,403 MB	(152 GB)

#### 6.4.4.2 Memory Costs

The memory consumption provided by the Java Virtual Machine after training the several classifiers for one epoch is depicted in Table 6.6. Note that these sizes include the overhead caused by the virtual machine and the machine learning framework.<sup>57</sup>

MLPP already consumes more memory than the dual variant for the first dataset with 200 classes. For the 400 classes of the *directory code* view the algorithm requires almost 2 GB, while DMLPP is able to compress the same information into slightly more than 200 MB. Remind that as expected MLPP is not applicable to *EUROVOC*. A simple estimation based on the number of base classifiers, number of features and bytes per float variable results in 152 GB of memory.

Another remarkable fact is that the memory requirement of DMLPP is comparable to that of the one-per-class algorithms: for the smaller datasets we obtain an overhead of only 50 MB and for the bigger *EUROVOC* view it requires only double of the memory, although representing a quadratic number of base classifiers. On the other hand, a view on the memory consumption of DCMLPP reveals that great part of the space for MMP/BR and DCMLPP is caused by the overhead of the JVM and the machine learning framework (instances and class mappings in memory, extensive statistics, etc.). If we compute the core memory requirements of BR by subtracting DMLPP's value from DCMLPP's for *EUROVOC*, we obtain 161 MB. In consequence we obtain a general overhead of 981 MB and thus an actual memory consumption of 261 MB for DMLPP. These values appear to be realistic after a simple estimation. In the same way we can compute the expected consumption for (Q)CMLPP by just adding 161 MB to MLPP's numbers.

Note also that the memory requirements of BR/MMP and MLPP strongly depend on the number of features  $a$ , while DMLPP's sensitivity to the sparse amount  $a'$  is quite independent on the original size  $a$ . In fact, one reason for reducing the feature dimensionality to 5000 by feature selection was to enable the primer variants, especially MLPP, to tackle the tasks. Assuming that the more than 200,000 original features encountered (cf. Section 6.1) were used, MMP/BR would require 40 times more space than given in Table 6.6 (1600 times more for MLPP).

<sup>57</sup> We used the WEKA framework (Witten and Frank 2005), but we adapted it so that it maintains a copy of a training instance in memory only when necessary for the incremental updating.

---

## 6.5 Related Work

Pouliquen and Steinberger et al. have already experimented with the EUROVOC thesaurus for cross-lingual similarity computation of documents using the associated EUROVOC keywords as substitute attributes (Steinberger et al. 2002), and for automatic keyword assignment (Pouliquen et al. 2003). The latter work basically follows a similar setting to our work. The authors use different refined text and linguistic processing techniques and statistical computations in order to return for a document a (manually limited) list of associated lemmas from the EUROVOC thesaurus. They obtained recall and precision values of around 40 to 50% on one test set and around 65% on a smaller different set of documents. Two human specialists judged manually the appropriateness of the returned descriptors in the latter case. However, these results are not comparable to those in this paper since a different set of document was used (also text from the EU, but from a different resource), resulting also in a different number of EUROVOC descriptors used, namely around 2900.

An also relatively large dataset with 1093 classes and 2802 texts was used in (Montejo Ráez et al. 2004). The authors introduced an algorithm that decomposes a multilabel problem into binary problems such as BR but with the peculiarity that it is able to discard base classifiers that perform poorly on their classes. The idea behind is the great unbalanced distribution of classes. The authors concentrate on more common classes and ignore infrequent ones as they do not influence the overall result in a great degree. The obvious drawback is the percentage of uncovered classes, that reaches up to 65%.

Another interesting aspect in the work of Montejo Ráez et al. is the reweighting of training examples in order to balance out the subproblems, since the unbalanced distribution and BR decomposition lead to very unbalanced subproblems. In our opinion, this problem is not too severe for the pairwise decomposition, since it does not compare one class against the accumulation of all remaining examples, achieving on average a more balanced factor between positive and negative examples. However, (D)MLPP is potentially able to use more advanced perceptron variants that also take into account unbalanced classes (Crammer et al. 2006, Li et al. 2002). Also, since we mainly evaluate the ranking quality, MMP and BR should not be discriminated by unbalanced classes in our experiments.

Tsoumakas et al. (2008) processed a dataset from the Web 2.0 with almost 1000 classes. Their hierarchical approach HOMER is the base of Chapter 7. The very recent technique of label space transformation was also successfully applied to this dataset (Hsu et al. 2009b, Tai and Lin 2010) and even to datasets containing up to 4000 classes (Bi and Kwok 2011). A short overview of this approach is given in Section 2.8.6.

However, Rubin et al. (2011) presented the first work (to appear soon) facing the challenging *EUR-Lex eurovoc* dataset. They adapted latent Dirichlet allocation (LDA) topic models approach to multilabel classification. LDA assumes that each label (*topic* in the terminology of LDA) is given as a distribution over words and a document is sampled from a mixture of label distributions. More specifically, each word in a document is assumed to be sampled from one label which was previously sampled for the document. In



contrast to the originally unsupervised LDA, these distributions are estimated from the mappings between documents and labels in the training data. In addition, Rubin et al. perform an unsupervised LDA on top of the labels in order to capture label dependencies and assume that the labels itself were sampled from a distribution over labelsets. These labelsets are not taken from the training data, but LDA computes a predetermined number  $k$  of them ( $k = 200$  in their experiments), hence we could interpret this process as a (probabilistic) multilabel clustering over the labels. During testing a document, one label is chosen for each word. The resulting label distribution induces a distribution over the labelsets, which subsequently influences the label distributions at the word-level. The process is repeated until a stable final label distribution is obtained. The authors indicate the training complexity as  $\mathcal{O}(ad + mdk)$  and testing with  $\mathcal{O}(a(n + k))$ , which however does not include convergence iterations.

A direct comparison with published results is always problematic, since a different feature selection, feature representation and train-test splits may have been used. However, the similar experimental settings may reveal a tendency, which of course has to be handled with care.<sup>58</sup> DCMLPP behaves favorably in terms of all ranking losses except RANKLOSS and MARGIN. However, for the latter two metrics the difference is clear. This results from the fact that RANKLOSS and MARGIN are more susceptible to outlier predictions, i.e. a single relevant label which is missed and predicted at a rather low position can excessively influence the overall score. From the comparison to the basic LDA version which does not induce labelset models on top (MARGIN = 708), it becomes obvious that the labelset models ensure that a label, which is itself predicted as rather unlikely by basic LDA but which was often observed together in the training set with the most likely labels in the predicted ranking, is pushed on a higher position. DCMLPP's prediction, in contrast, completely relies on the accumulation of preference statements for an individual label. Ranking correcting techniques or alternative aggregation strategies (see Section 3.5.5 for further explanations) could help the pairwise classifier to surmount this shortcoming.

## 6.6 Summary

In this chapter, we introduced the *EUR-Lex* text collection as a promising test bed for studies in text categorization. Among its many interesting characteristics (e.g., multilinguality), our main interest was the large number of categories, which is one order of magnitude above other frequently studied text categorization benchmarks, such as the *Reuters-rcv1* collection.

On the *EUROVOC* classification task, a multilabel classification task with 4000 possible labels, the DMLPP algorithm, which decomposes the problem into training classifiers

<sup>58</sup> An older version of *EUR-Lex* was used (Loza Mencía and Fürnkranz 2008a), which contains slightly more documents (cf. Section 6.1). The bag of words on the same cross-validation splits were pre-processed slightly differently. In the following, we reproduce the scores of the LDA-classifier, the values for DCMLPP trained in two epochs (Table 6.3) and the performance on the older dataset from Loza Mencía and Fürnkranz (2008a): IsErr 97.2/96.75/96.6, OneErr 32/28.01/29.5, AvgP 51.1/52.72/52.3, F1<sub>|p|</sub> 47.17/48.04/–, RANKLOSS 1.77/2.398/2.5, MARGIN 269/387.51/397.

---

for each pair of classes, achieves an average precision rate of slightly more than 50%. Roughly speaking, this means that the (on average) five relevant labels of a document will (again, on average) appear within the first 10 ranks in the relevancy ranking of the 4,000 labels. This is a very encouraging result for a possible automated or semi-automated real-world application for categorizing EU legal documents into EUROVOC categories.

This result was only possible by finding an efficient solution for storing the approx. 8,000,000 binary classifiers that have to be trained by this pairwise approach. To this end, we showed that a reformulation of the pairwise decomposition approach into a dual form is capable of handling very complex problems and can therefore compete with the approaches that use only one classifier per class.

It was demonstrated that decomposing the initial problem into smaller problems for each pair of classes achieves higher prediction accuracy on the *EUR-Lex* data, since DMLPP substantially outperforms all other algorithms. This confirms the previous results in Chapter 4 and 5 of the non-dual variant on the large text classification testbed Reuters Corpus Volume 1 and also on a wide range of non-textual datasets. The dual form representation allows for handling a much higher number of classes than the explicit representation, albeit with an increased dependence on the training set size. Despite the improved ability to handle large problems, DMLPP is still less efficient than MMP, especially for the *EUROVOC* data with 4000 classes. However, in our opinion the results show that DMLPP is still competitive for solving large-scale problems in practice, especially considering the trade-off between runtime and prediction performance.

The experiments in this chapter confirm the main statement about the adaptation of the QVoting technique introduced in Chapter 5, namely that the technique permits to reduce the amount of perceptron predictions of the MLPP algorithm during the classification of the *subject matter* and *directory code* views to a level competitive to BR/MMP. However, the processing of the almost 4000 classes of *EUROVOC* is out of scope, making it still necessary to use techniques such as the Dual MLPP or hierarchical decomposition techniques as introduced in the next chapter.

For future research, we see space for improvement in using hybrid variants in order to further reduce the computational complexity and in the use of more advanced perceptron training variants or SVMs (cf. Section 6.2.2) in order to further improve the prediction quality.



---

## 7 Hierarchical Model Efficiency and Scalability

The former chapter presented an approach that is potentially able to pairwise decompose challenging problems with almost arbitrary large number of labels. However, even with the trick of dual reformulation, problems of this type of high dimensionality are hard to tackle. Furthermore, the limitation on perceptron or support vector based learners and the sensitivity to training size and label density have to be taken into consideration if we want to employ DMLPP. For the normal variants the main bottleneck remains to be the quadratic memory consumption with respect to the number of labels.

Recently (Tsoumakas et al. 2008) introduced the *HOMER* approach which decomposes the problem into a hierarchy of simpler multilabel problems, where each subproblem uses a reduced number of possible labels. The hierarchical structure of the labels is obtained by applying recursive clustering to the initial set of labels. In the cited work HOMER was applied to a key tagging problem with almost 1000 labels. The empirical evaluation showed that it was able to outperform conventional BR with respect to predictive accuracy and, especially, classification time, which in turn indicates a sub-linear dependency on the number of labels.

Similar positive results can be expected from using pairwise ensembles for solving the subproblems generated by HOMER. More importantly, HOMER decomposes the original problem into considerably smaller multilabel problems with a maximum number of labels. This would have a direct impact on the required memory space and would enable to apply the pairwise technique to tasks of almost any arbitrary large size.

In fact, our analytical and experimental results indicate that HOMER is able to improve the classification performance, training time, and classification time for the calibrated ranking approach on four datasets of mid- and high-range size. Additional evaluations with binary relevance decomposition reveals that the supremacy on predictive quality is maintained while the computational costs are on the same level or even clearly fall below.

However, we must bear in mind that we partly abandon the paradigm of pairwise preference learning. By applying the pairwise decomposition on only subsets of the original class set we renounce to codify a great part of the preference information between labels inherent in the original data. Nonetheless, the advantages are convincing and outweigh the disadvantages of relaxing the pairwise setting particularly on large tasks.

The following section informally and formally describes the HOMER algorithm. The base study by Tsoumakas et al. (2009b) is extended by a detailed analysis of the computational costs in Section 7.2. The evaluation of the usage of different hierarchy clustering approaches and the comparison between the different possible combinations of HOMER,

binary relevance and calibrated label ranking are presented in Section 7.3 and summarized in Section 7.5.

## 7.1 HOMER: Hierarchy of Multilabel Classifiers

The HOMER algorithm was introduced by Tsoumakas et al. (2008) and allows to use base multilabel learners yet being sensible to the number of labels by decomposing the original problem into a tree of multilabel subproblems: a predetermined number of labels are joined to one metalabel, which is in turn one possible label in the parent multilabel subproblem. During prediction, the multilabel classifier at each inner node starting from the root is queried and the children nodes are visited for which the metalabel was predicted. The leaves represent the labels from the original problem.

### 7.1.1 Training

More formally, the algorithm works as follows. We assume at this point that a hierarchy  $\mathcal{H}$  on  $\mathcal{L}$  and metalabels  $\mathcal{M} = 2^{\mathcal{L}}$  is already given.  $\mathcal{H} \subset \mathcal{M}^2 \cup \mathcal{M} \times \mathcal{L}$  is defined as a partial order that spans a rooted directed tree with a compositional semantic (cf. Section 2.5.6), i.e.

$$\mu_u \succ_{\mathcal{H}} \mu_v \Leftrightarrow (\mu_u, \mu_v) \in \mathcal{H} \Leftrightarrow \mu_v \subset \mu_u \quad \bigcup_{\mu_u \succ_{\mathcal{H}} \mu_v} \mu_v = \mu_u \quad \bigcup_{\mu_u, \mu_w \succ_{\mathcal{H}} \mu_v} \mu_u \cap \mu_w = \emptyset \quad (7.1)$$

We denote the metalabel at root node 1 by  $\mu_1 = \mathcal{L}$  and the children metalabels at nodes  $1.i$ ,  $1 \leq i \leq k' \leq k$  by  $\mu_{1.i}$ , where  $\mu_1 \succ_{\mathcal{H}} \mu_{1.i}$ , and so forth until the original labels are reached at the leaves,  $\lambda_i = \mu_{1\dots}$ . The ramification  $k'$  is bounded by the maximal number  $k$  of (meta)labels that can be contained in one metalabel. Each inner node  $1, 2, 3 \dots$  is associated to a multilabel classifier  $h_1, h_2, h_3 \dots$ .

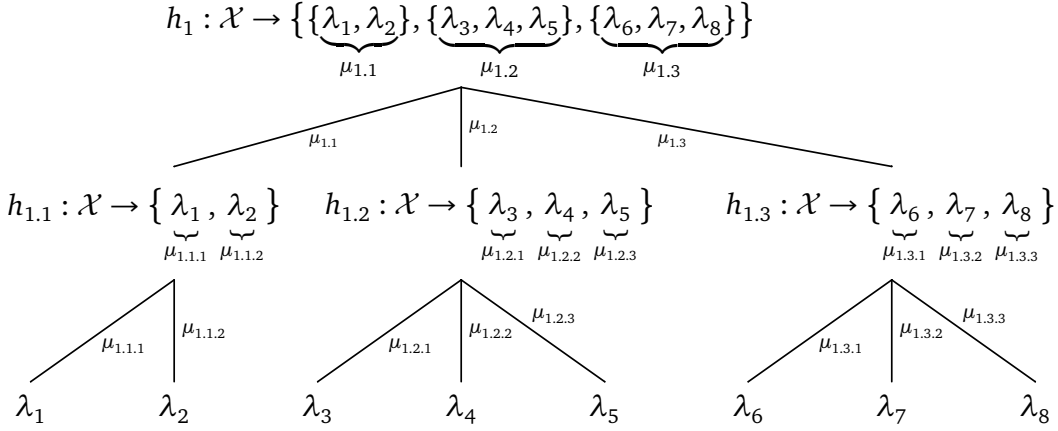
A training example  $(\mathbf{x}, P)$  arrives at first at the root node, where it is used to train the classifier  $h_1 : \mathcal{X} \rightarrow \{\mu_{1.1}, \dots, \mu_{1.k}\}$ .  $\mathbf{x}$  serves as positive example for metalabels  $P^1 := \{\mu_{1.i} \mid P \cap \mu_{1.i} \neq \emptyset\}$ . The training example is then passed to all children nodes  $1.i$  with positive metalabels  $\mu_{1.i} \in P^1$ .

The classifiers at the inner nodes are trained similarly. Formally, the training set for a classifier  $h_z : \mathcal{X} \rightarrow \{\mu_{z.1}, \dots, \mu_{z.k}\} \subset 2^{\mu_z}$  at node  $z = 1..i$  is given as follows:

$$\mathcal{T}_{rain}^z := \langle (\mathbf{x}, P^z) \mid P \cap \mu_z \neq \emptyset \rangle \quad P^z := \{\mu_{z.i} \mid P \cap \mu_{z.i} \neq \emptyset\} \quad (7.2)$$

Hence, at each node  $z$  the data is filtered so that only examples pass that are annotated with at least on of its own label  $\mu_z$ .

An example for a hierarchy of eight original labels is illustrated in Figure 7.1. Imagine an example  $\mathbf{x}$  with  $P = \{\lambda_1, \lambda_2, \lambda_7\}$ . For  $h_1$  the example would be positive for  $\mu_{1.1}$  and  $\mu_{1.2}$ . Hence,  $h_{1.1}$  would see the example as  $P^{1.1} = \{\mu_{1.1.1}, \mu_{1.1.2}\}$  and  $h_{1.3}$  as positive for  $\mu_{1.3.2} = \lambda_7$ . The classifier  $h_{1.2}$  would not receive it.



**Figure 7.1:** A sample hierarchy of multilabel classifiers. An example is forwarded to a child node if the metalabel  $\mu_z$  at the edge coincides with  $P$ , i.e.  $\mu_z \cap P \neq \emptyset$ . Similarly, an edge  $\mu_{a,z}$  is followed during prediction if  $\mu_{a,z} \in \hat{P}^a$ ,  $\hat{P}^a = h_a(\mathbf{x})$  (example based on Tsoumakas et al. 2008).

### 7.1.2 Predicting

The prediction works similar. A test example  $\mathbf{x}$  is classified by  $h_1$  and only the predicted nodes  $z$  with  $\mu_z \in h_1(\mathbf{x})$  are visited. The remaining levels are stepped through recursively until possibly predicting the original labels at the leaves. Formally, the following labelset is predicted

$$\hat{P} = \{\lambda_i \mid \bigwedge_{\mu_z \in \hat{P}_1} \lambda_i \in h_z(\mathbf{x})\} \quad (7.3)$$

Returning to our example above, assuming a consistent base learner, the example would be predicted as  $\mu_{1.1}$  and  $\mu_{1.3}$  by  $h_1$ . Hence, only  $h_{1.1}$  and  $h_{1.3}$  have to be further consulted, since neither  $\lambda_6, \lambda_7$  or  $\lambda_8$  are in  $h_1(\mathbf{x})$ . However, it holds that  $\lambda_1, \lambda_2 \in h_{1.1}(\mathbf{x})$  and  $\lambda_7 \in h_{1.3}(\mathbf{x})$  and so these would be the predicted classes  $\hat{P}$ . Note that it could happen that the root  $h_1$  predicts  $\mu_{1.1}$ , but that  $h_{1.1}$  returns the empty set. In this case, the evaluation stops at the inner node and no label from this sub-tree is predicted.

### 7.1.3 Hierarchy Construction

Before the actual training phase starts, HOMER has to create the hierarchy tree  $\mathcal{H}$ . This is done recursively in a top-down depth-first fashion starting with the root. At each node  $\mu_z$ ,  $k$  child nodes are first created using a clustering algorithm (see below). In case  $|\mu_z| < k$ , the number of children is set to  $k' = |\mu_z|$ .

The main issue in the former process is how to distribute the labels of  $\mu_z$  to the  $k$  children. We argue that labels should be *evenly* distributed to  $k$  subsets in a way such that labels belonging to the same subset are as *similar* as possible. Such a task can be

thought of as clustering with the additional constrain of equal cluster size. It has been considered in the past in the literature, under the name *balanced clustering* (Banerjee and Ghosh 2006). In the work of Tsoumakas et al. (2008), a new balanced clustering algorithm named balanced  $k$ -means has been proposed for HOMER, which guarantees that the clusters will be of exactly the same size. The base is a recursive  $k$ -means on the label columns of the output matrix  $(\mathbf{y}_i)_i$ ,  $1 \leq i \leq m = |\mathcal{T}_{rain}|$ .

We will denote this approach with  $B$  for balanced clustering. The other two approaches considered are random and even distribution ( $R$ ) of the labels to the children nodes and clustering ( $C$ ) using the expectation minimization algorithm (as implemented in WEKA, Witten and Frank 2005).

The justification for preferring similarity-based distribution is that if similar labels of a node  $\mu_z$  are placed in the same subset, then only a few (ideally just one) metalabels  $\mu_{z,1}, \mu_{z,2} \dots$  will be predicted and thus the remaining sub-trees will not be activated. This will lead to reduced costs during the operation and testing of HOMER. Another expected benefit is that each child node will probably contain less training examples. The complexity of HOMER will be discussed in more detail in Section 7.2. The justification for preferring an even distribution is that the multilabel classifiers at each node will deal with a more balanced distribution of positive examples for each metalabel. This is expected to lead to improved predictive performance.

We could consider HOMER as the combination of two components: 1) an algorithm that constructs a hierarchy on top of the labels of a multilabel dataset (cf. references on hierarchy extraction in Section 2.5.6), and 2) a generalization of the well-known Pachinko-machine hierarchical classification algorithm (Koller and Sahami 1997) to the multilabel case.

In the work of Tsoumakas et al. (2008), HOMER, when using the well-known binary relevance classifier as the base multilabel classifier in each internal node, has proven to outperform BR in terms of quality of prediction and, especially, classification time. In this study, we particularly compare to the calibrated pairwise label ranking approach (cf. Section 3.4.5) as multilabel classifiers at the nodes of HOMER.

We will use the following abbreviations throughout this chapter in order to indicate the different possible combinations.  $H+BR$  will denote the combination of HOMER with binary relevance as decomposition strategy for the multilabel subproblems. Correspondingly,  $H+CLR$  or  $H+QCLR$ , respectively, will denote the combination with the (QVoting) calibration label ranking approach. Note that  $H+CLR$  and  $H+QCLR$  use the same model, hence we will preferably use  $H+CLR$  for indicating the training and  $H+QCLR$  for the prediction phase.

## 7.2 Computational Complexity

This section extends the previous analysis by Tsoumakas et al. (2008) particularly regarding the combination of HOMER with the decompositive approaches. We refer to this original analysis for the detailed costs of the clustering approach.

For the sake of simplification of the analysis, we will assume that we have a perfect  $k$ -ary tree of  $N + 1$  levels and hence  $k \cdot k \cdot \dots = k^{N+1} = n$  leafs. Such a tree has  $k^{i-1}$  nodes at the  $i$ -th level, the root being the first level, thus in total  $\sum_{i=1}^N k^{i-1} = \frac{k^N}{k-1} = \frac{n-1}{k-1}$  inner nodes. Any other tree with  $N$  levels and maximal ramification factor  $k$  will at most require the costs of using the perfect tree. Further, we will need the average  $d$  of the label cardinalities  $|P|$  and refer to previous analysis of BR, LPC and (Q)CLR in Sections 3.4.6 and 5.2.

Estimations are given in terms of (binary) classifiers for the memory requirements, number of total examples for training and number of (binary) classifier evaluations for prediction. Remind that the actual bytes and seconds, and especially the ratios given, may deviate from the estimations given here particularly if the binary base classifier used has not a linear behavior in number of training examples for training and is not constant for memory and testing time (cf. Section 3.4.6.3). Particularly, the following analysis states training (and testing) costs on per example basis and not in terms of the training size. A summary of complexities and relationships between the approaches are given in Landau big  $\mathcal{O}$  notation in Table 7.1.

### 7.2.1 Memory

The number of inner nodes already determine the required number of multilabel classifiers. For BR this amount has to be multiplied with  $k$  in order to obtain the number of binary classifiers  $k \frac{n-1}{k-1} = \mathcal{O}(n)$  that have to be maintained in memory. This is less then for pairwise decomposition, which will require additional  $k(k-1)/2 \frac{n-1}{k-1} = k(n-1)/2$  binary classifiers.

If we compare the requirements to those of the bases BR and CLR, we see a twofold result. For a fixed  $k$ , the comparison between H+BR and BR results in a ratio of  $\frac{k(n-1)}{n(k-1)}$  in favor of BR, but which becomes only nearly  $k/(k-1)$  for increasing  $n$ . For the additional pairwise classifiers the ratio is  $\frac{k(n-1)}{n(n-1)} = k/n$ , which is a clear and important reduction. The ratio between conventional CLR and BR is reduced from  $(n+1)/2$  to

$$\frac{k(n-1)}{k-1} / \frac{k(k+1)(n-1)}{2(k-1)} = \frac{2}{k+1} \quad (7.4)$$

for the combination with HOMER. Approximately the same relationship to normal BR applies.

### 7.2.2 Training

During training the root processes all training examples. A training instance (with a non-empty labelset) is passed to at least one children and at most to  $\min(|P|, k)$  children, depending on the clustering algorithm used and fortuity. At the subsequent levels, an example will be reached over to at most  $\min(|P|, k^i)$  children nodes on the  $(i+1)$ -th

**Table 7.1:** Complexity comparison of BR and CLR and their combinations with HOMER. The row captions indicate the base decompositions BR and CLR and their asymptotic complexities, whereas the column caption denote the respective combinations with HOMER and their costs. In the cells we find the ratio between the HOMER variants and the base decompositive approaches with respect to the column and row captions. The costs are given in terms of the metering indicated in the sub-tables caption.

(a) total number of binary base classifiers				(b) preferences learned per training example			
		H+BR	H+CLR			H+BR	H+CLR
		$\mathcal{O}(n)$	$\mathcal{O}(kn)$			$\mathcal{O}(kd \log_k n)$	$\mathcal{O}(kd \log_k n)$
BR	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(k)$	BR	$\mathcal{O}(n)$	$\mathcal{O}(\frac{d \log n}{n})$	$\mathcal{O}(\frac{d \log n}{n})$
CLR	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(k/n)$	CLR	$\mathcal{O}(dn)$	$\mathcal{O}(\frac{\log n}{n})$	$\mathcal{O}(\frac{\log n}{n})$

(c) binary base classifier evaluations for one prediction			
		H+BR	H+QCLR
		$\mathcal{O}(kd \log_k n)$	$\mathcal{O}(kd \log n)$
BR	$\mathcal{O}(n)$	$\mathcal{O}(\frac{d \log n}{n})$	$\mathcal{O}(\frac{kd \log n}{n})$
QCLR	$\mathcal{O}(dn \log n)$	$\mathcal{O}(\frac{k}{n \log k})$	$\mathcal{O}(\frac{k}{n})$

level. We can hence assume that an example will be used  $\leq |P|$  times at each level except the last one (where there are no classifiers to train).

The multilabel classifiers at the inner nodes of a HOMER ensemble will hence process one single training examples maximally  $1 + |P|(N - 2) = \mathcal{O}(d \log_k n)$  times. The lower bound is also determined by the depth of the tree, since at least  $N$  nodes have to be visited (assuming  $|P| \geq 1$ ).

For the combination of HOMER and the binary relevance decomposition, each example at each affected BR classifier is further replicated for each of the  $k$  binary base classifiers, resulting in  $\mathcal{O}(kd \log_k n)$  instances processed at the lowest classifier level. The same paths trough the tree means to use each example  $|P|(k - |P|)/2 + (N - 1)(k - 1)|P| < |P|k + Nk|P| = \mathcal{O}(kd \log_k n)$  times in the pairwise base classifiers, which is even slightly less than for BR in many cases, albeit we have to add the costs of the BR classifiers itself to CLR. However, this is theoretically not the worst possible strategy for CLR. Certainly, training an individual CLR classifier becomes most expensive when an instance is mapped to  $k/2$  children meta-labels, which results in  $k^2/4$  preferences learned. However, after at most  $\log_{k/2} |P|$  levels the label cardinality becomes one. It is hence not obvious how to determine the worst case for the combination with CLR.

For both, in the best case the  $|P|$  relevant labels are divided at only one inner node at level  $N$  (assuming  $|P| \leq k$ ). Hence both complexities are bounded from below by

$kN = k \log_k n$  (BR) and  $(k - 1)(N - 1) + |P|(k - |P|)/2 < k(\log_k n + |P|)$  (pairwise preferences learned).

Comparing H+BR to BR, we achieve a ratio of at worst  $\mathcal{O}(kd \log_k(n)/n) = \mathcal{O}(\frac{kd \log n}{n \log d})$  in the number of training examples. Assuming a fixed  $k$ , the ratio rapidly decreases asymptotically to  $d \log(n)/n$  with increasing  $n$ . The same applies to the comparison of H+CLR and BR. Consequently, the relationship between the pairwise preferences learned by H+CLR and CLR decreases even faster, since  $\mathcal{O}(kd \log_k(n)/(dn)) = \mathcal{O}(\log(n)/n)$ .

Certainly, it seems recommendable to analyze the *average* case, where we assume that the problems at the nodes have similar characteristics as the original multilabel problem. Basically, this corresponds to analyzing the randomized clustering strategy. We can expect that more sophisticated strategies will ideally perform better and at worst asymptotically approach the average case. Unfortunately, the analysis of this case is not as trivial as it may appear at the first sight, since it requires an elaborated probabilistic and stochastic modeling. Therefore, and because of the generally coarse estimation this would mean for clusterings others than randomized, we refer to the in our eyes in this case more practical empirical comparison of the costs in Section 7.3 and leave the formal analysis for further research.

### 7.2.3 Predicting

As was explained in Section 7.1, the prediction phase is very similar to the training phase. Assuming that the base multilabel classifiers predict roughly the same number of labels than in the training data, the same amount of inner nodes are visited during testing an instance. For the combination with BR this means absolutely the same estimations as for training.

For the pairwise classifiers however, the costs are substantially reduced due to the decreased size  $k$  of the subproblems. The ratio between H+CLR's and CLR's base classifier evaluations amounts to  $\mathcal{O}(k^2 \cdot d \log_k(n)/n^2) = \mathcal{O}(d \log(n)/n^2)$  for a fixed  $k$ . Our estimation of the training time consisted in assuming a maximal ramification of  $|P|$  at the root and hence single-label assignments in the consecutive  $|P|$  paths to the leaves. Under these circumstances, we estimate  $\mathcal{O}(k \log k \cdot d \log_k n)$  evaluations performed by using QVoting in HOMER compared to the well known  $\mathcal{O}(dn \log n)$ . The ratio between H+QCLR and QCLR is hence increased to now  $\mathcal{O}(k/n)$ . The comparison to BR is slightly worse but similar, since H+QCLR requires  $\mathcal{O}(kd \log(n)/n)$  times the base classifier evaluations of BR. H+QCLR should only take an order of  $\mathcal{O}(\log k)$  times more computations than its counterpart H+BR.

## 7.3 Evaluation

In this section, after the presentation of the experimental setup, we will discuss the effect of the several parameters of HOMER and then compare it in terms of training time, classification time and predictive performance against its base multilabel classifiers.



**Table 7.2:** Name, number of examples used for training and testing, number of features and labels, label cardinality and density, and number of distinct labelsets for each dataset used in the experiments.

name	examples		features	labels	cardinality	density	distinct labelsets
	train	test					
<i>hifind</i>	16452	16519	98	632	37.304	6.0%	32734
<i>eccv2002</i>	42379	4686	36	374	3.525	0.9%	3175
<i>jmlr2003</i>	48859	16503	46	153	3.071	2.0%	3115
<i>mediamill</i>	30993	12914	120	101	4.376	4.3%	6555

### 7.3.1 Setup

We conducted experiments on four large multilabel datasets with at least 100 labels and 10000 training examples. The first one, *hifind*, contains 32769 music titles annotated on average with 37 from 632 different labels. The two next datasets *eccv2002* and *jmlr2003* are from the image classification domain, whereas the last one deals with video material. All datasets were already described in more detail in Section 2.9.1. Table 7.2 summarizes the key characteristics of the datasets.

The experiments were conducted using the Mulan library of algorithms for multilabel learning (Tsoumakas et al. 2011b). As base classifier we used the decision tree learner J48 with standard settings, which is an implementation of C4.5 included in the WEKA framework (Witten and Frank 2005).

Note that only BR is able to predict label rankings in the basic versions, so we can only evaluate the bipartition quality. The effectiveness of all algorithms is evaluated with (label and example-based) micro-averaged recall, precision and F1 (cf. Section 2.7.3). We also evaluate the efficiency of all algorithms based on their run time (for training and classification).

### 7.3.2 Results of HOMER with QCLR

This section presents and discusses the results of using HOMER together with QCLR as the multilabel algorithm for building models at each internal node of the hierarchy. We experimented with 8 different numbers of partitions (i.e.,  $k$  ranges from 3 to 10) and 3 different methods for partitioning the set of labels at each internal node: 1) random and even distribution (R) of the labels to the children nodes, 2) clustering (C) using the expectation minimization (EM) algorithm (as implemented in WEKA), and 3) balanced clustering (B) using the algorithm introduced by Tsoumakas et al. (2008). In addition to HOMER with QCLR as multilabel classifier we ran the experiments using HOMER with BR and also using the plain algorithms BR and QCLR without HOMER. In the following graphs the combinations of HOMER, QCLR and the respective partitioning approaches are denoted CLR-R, CLR-B, and CLR-C, respectively. The combinations with BR are called BR-R, BR-B, BR-C, respectively.



---

### 7.3.2.1 Training Time

Figure 7.2 shows the training time of the HOMER variants in seconds. We would expect that the training time of the random partitioning variant should be less than that of the balanced clustering variant, since they both deal with the same number of labels and create and train the same number of multilabel classifiers, but balanced clustering needs some additional time to distribute the labels according to similarity as well.<sup>59</sup> However, this is clearly noticed only in *eccv2002*. In *jmlr2003* there is no clear winner for all numbers of partitions, while in *mediamill* and *hifind* we notice that the balanced clustering approach requires less time, independently of the multilabel learning algorithm that is used (BR or CLR) and the number of partitions.

These results can be explained by the following observation. As clustering is based on the values of the labels, the children produced with balanced clustering will contain labels that typically appear or do not appear together. This in turn means that more examples of the parent node will be filtered, leading to a reduced number of training examples. This was also observed by Tsoumakas et al. (2008). Here, we notice that the gains in training time are higher for CLR compared to BR. This is an expected result based on the previous observation, because CLR trains its binary classifiers only on those examples where the values of the corresponding labels differ.

One issue that still needs to be explained is how this behavior is affected by the different datasets. In this direction, we notice that the gains in training time seem to be correlated with the density of the dataset. The reason, again based on the previous observation, is that the lower the number of label appearances with respect to the number of labels (density), the lower the gains that can be achieved by clustering co-occurring labels together.

Concerning the plain clustering partitioning method, we notice that it is clearly the worst one in terms of training apart from the *mediamill* dataset. The plain clustering method requires more time to perform the clustering as it is based on the expectation maximization algorithm. *mediamill* is also the smallest dataset, where it seems that the time required for clustering does not surpass the gains from the clustering process. This is why plain clustering appears to be better than random partitioning, especially for CLR. The loss in performance is more evident in *eccv2002* and *jmlr2003* due to the lower density.

### 7.3.2.2 Testing Time

Figure 7.3 shows the testing times of the HOMER variants in seconds. Here the results are not as clear as in the case of the training time. Apart from the *jmlr2003* dataset, it seems that balanced clustering leads to less testing time compared to random partitioning irrespectively of the multilabel learning algorithm. Also plain clustering seems to be worse than the rest of the partitioning methods in *eccv2002* and *jmlr2003* for most of

---

<sup>59</sup> Proper evaluation should separately measure the time to build the models and the time for balanced clustering. This should be considered in future works.

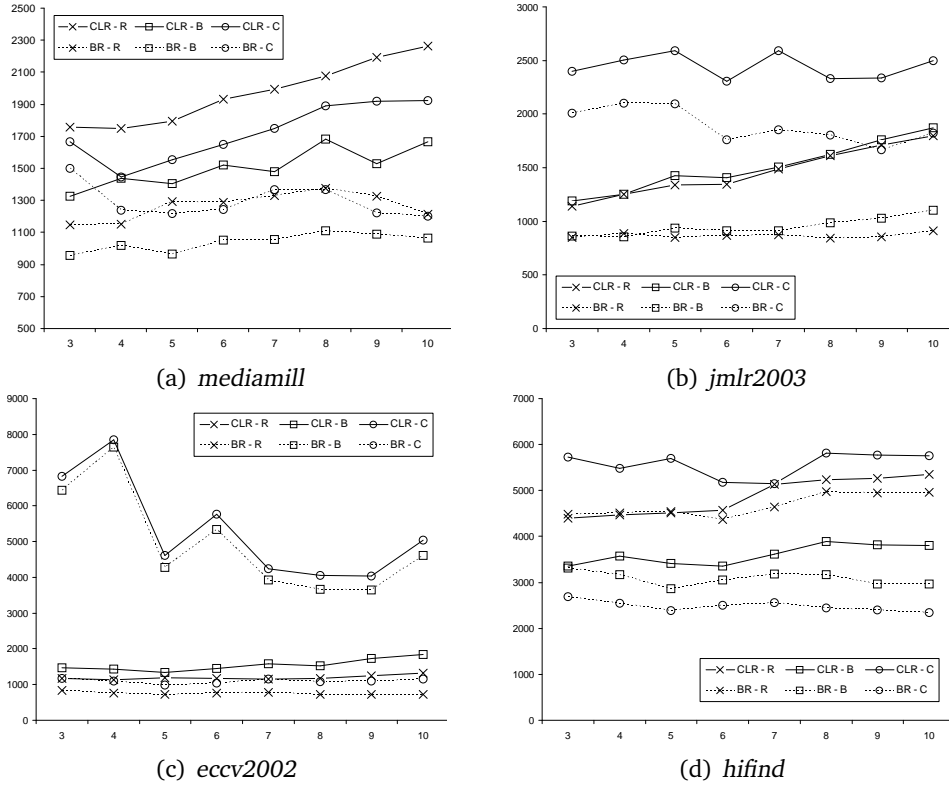


Figure 7.2: Training time over number of partitions for the six HOMER variants.

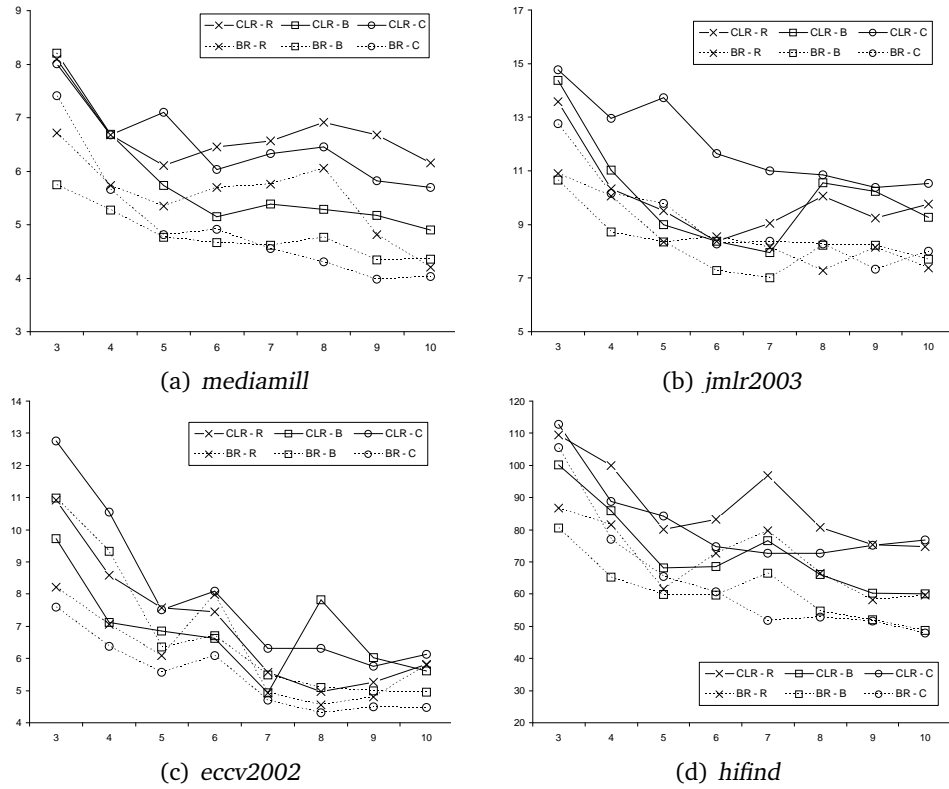


Figure 7.3: Testing time over number of partitions for the six HOMER variants.

---

the partition numbers. Finally, we could comment that the classification time seems to decrease with respect to the number of partitions probably due the smaller height of the tree ( $\log_k(n)$ ).

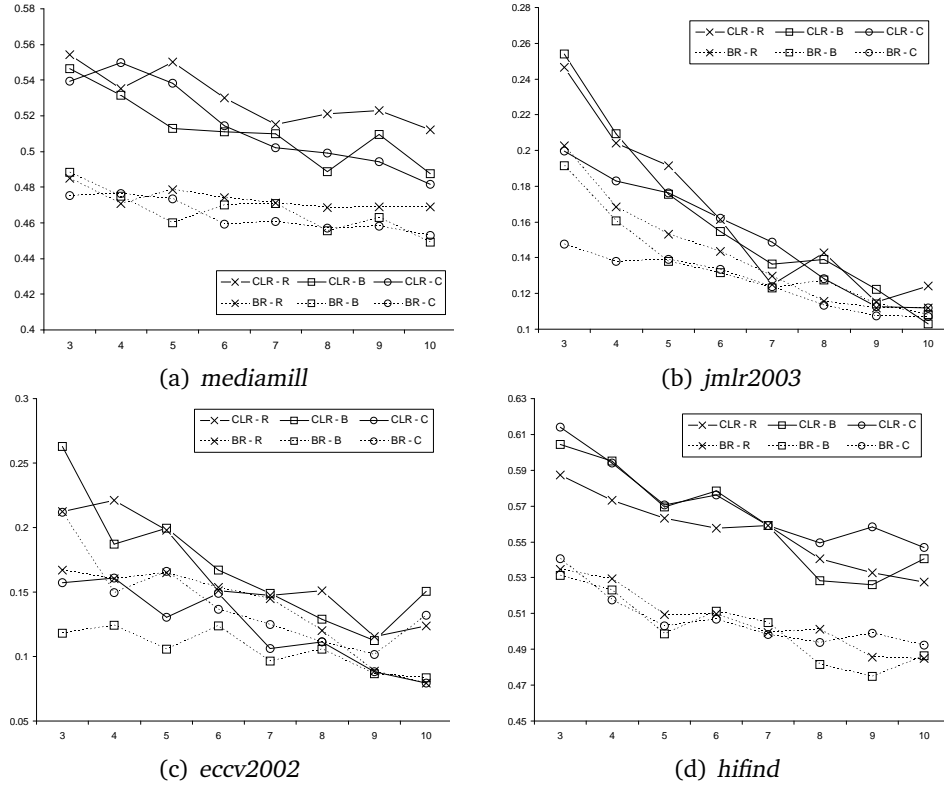
### 7.3.2.3 Predictive Quality

Figures 7.4 and 7.5 show the (micro-averaged) recall and precision results for the HOMER variants on all four datasets. We can see that recall decreases, while precision increases with the number of partitions, independently of the multilabel learner and partitioning method used. One potential reason for this behavior could be that smaller number of partitions lead to more general meta-labels that are more difficult to distinguish. Apparently this leads to a more relaxed prediction, so that at each inner node the multilabel classifier does predict more meta-labels and as a consequence more of the original labels, but with lower precision.

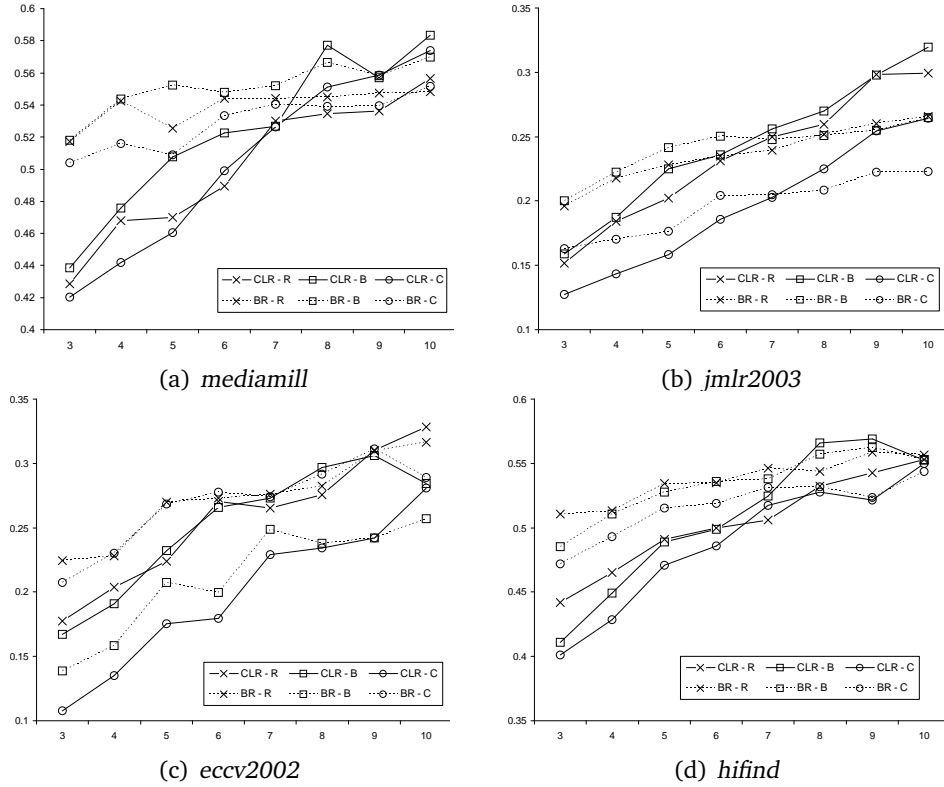
The recall of the CLR based HOMER variants seems to be larger than that of the BR based HOMER variants, irrespectively of the number of partitions. This is totally clear in *mediamill* and *hifind*, but less clear in *jmlr2003* and *eccv2002*, though it stills holds if we compare the two learners under the same partitioning method. As far as the partitioning method is concerned, there is no clear trend with respect to recall, while the plain clustering method seems to have the worst precision for both BR and CLR based HOMER.

The decrease in recall is stronger for CLR than for BR in the low density datasets *eccv2002* and *jmlr2003*. This means that in low density datasets, a small number of partitions favors the recall of HOMER with CLR. On the other hand the increase in precision is stronger for CLR than for BR in the high density datasets *mediamill* and *hifind*. This means that in high density datasets a large number of partitions favors the precision of HOMER. A potential reason is the fact that CLR underestimates the size of the predicted labelsets (cf. Section 4.6.9).

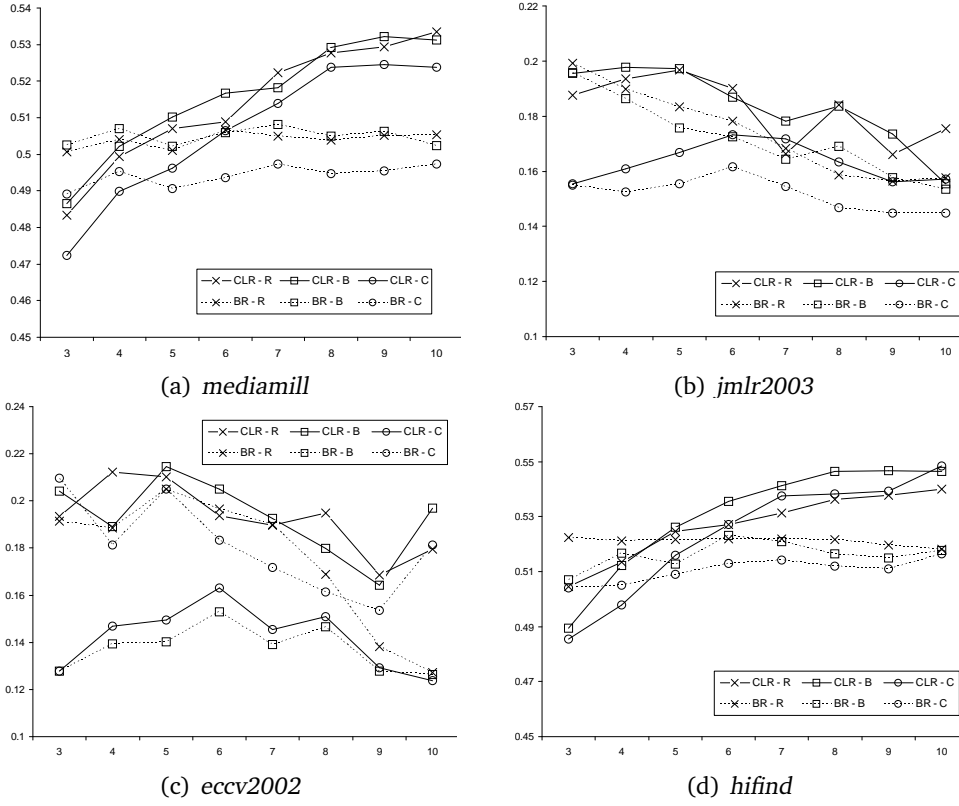
Figure 7.6 shows the micro-averaged F1 measure of HOMER for the datasets. As far as BR based HOMER is concerned no clear trend can be detected with respect to the number of partitions. With respect to the partitioning method, the plain clustering approach seems inferior to the rest, while no clear winner between balanced clustering and random partitioning can be announced. As far as CLR is concerned, as already outlined in the previous paragraphs, in low density datasets we notice a decrease of F1 with respect to the number of partitions, while in high density datasets we notice an increase of F1. We could therefore consider this as a guideline for selecting the number of partitions for HOMER with CLR based on the density of the dataset. Overall, the CLR based HOMER seems to be achieving better results for a larger percentage of different partition numbers, compared to the BR based HOMER. In terms of the partitioning method, the plain clustering approach seems inferior to the rest for both CLR and BR.



**Figure 7.4:** Micro recall over number of partitions for the six HOMER variants.



**Figure 7.5:** Micro precision over number of partitions for the six HOMER variants.



**Figure 7.6:** Micro F1 over number of partitions for the six HOMER variants.

### 7.3.3 Comparison of HOMER against its Base Classifiers

For the direct comparison of HOMER against the flat approaches in Table 7.3 we chose the configuration with balanced clustering and 10 partitions. Note that no results could be retrieved for CLR on the *hifind* dataset due to the high memory requirements. To circumvent this problem for problems with a large number of classes was a main objective of combining HOMER with CLR.

#### 7.3.3.1 Predictive Quality

It is especially interesting to observe the opposite behavior in terms of recall and precision of the different approaches. CLR shows the best precision performance with a large margin over the other algorithms on all datasets. On the other hand, its recall values are particularly low. This confirms previous results throughout this work that CLR does underestimate the size of the predicted labelsets (cf. Section 4.6.9).

Our results indicate that this is particularly true for datasets with a low label density such as *eccv2002*, where CLR returns only 3.84% of the correct labels, while 58.11% of the returned labels are actually correct, compared to the 36.58% by BR and around

**Table 7.3:** Bipartition measures and computational costs on the different datasets. Results for Binary Relevance (BR), (QVoting) calibrated label ranking (QCLR/CLR), HOMER with balanced clustering and 10 partitions and BR (H+BR), and HOMER with (Q)CLR (H+QCLR/CLR) are shown. Best values are indicated in bold.

Method	<i>mediamill</i>	<i>jmlr2003</i>	<i>eccv2002</i>	<i>hifind</i>	
BR	58.00 %	32.27 %	36.58 %	<b>59.43 %</b>	micro Prec
QCLR	<b>73.89 %</b>	<b>56.18 %</b>	<b>58.11 %</b>	–	
H+BR	56.98 %	26.48 %	28.91 %	55.31 %	
H+QCLR	58.35 %	31.93 %	28.43 %	55.26 %	
BR	44.79 %	9.85 %	7.42 %	45.73 %	micro Rec
QCLR	43.86 %	4.57 %	3.84 %	–	
H+BR	44.91 %	<b>10.81 %</b>	13.21 %	48.64 %	
H+QCLR	<b>48.77 %</b>	10.28 %	<b>15.07 %</b>	<b>54.06 %</b>	
BR	50.55 %	15.09 %	12.34 %	51.65 %	micro F1
QCLR	<b>55.04 %</b>	8.45 %	7.21 %	–	
H+BR	50.23 %	15.36 %	18.14 %	51.76 %	
H+QCLR	53.13 %	<b>15.55 %</b>	<b>19.70 %</b>	<b>54.65 %</b>	
BR	2413.40	2801.17	2701.32	4179.66	training time in s
CLR	7423.19	6542.51	7460.14	–	
H+BR	<b>1065.21</b>	<b>1101.61</b>	<b>1144.47</b>	<b>2345.39</b>	
H+CLR	1667.29	1871.00	1836.34	3801.53	
BR	<b>3.84</b>	<b>6.67</b>	5.47	50.47	testing time in s
QCLR	103.59	119.28	154.65	–	
H+BR	4.35	7.70	<b>4.48</b>	<b>48.77</b>	
H+QCLR	4.90	9.26	5.62	60.02	

28% by both HOMERs. On the other hand, on the *mediamill* dataset, where the density already ascends to 4.3% , CLR’s gain in precision seems to make up its low recall, thereby producing the highest average F1 value.

BR has a similar behavior of predicting relatively few labels with increasing number of labels. This is probably due to the greater imbalance of positive to negative examples for large problems, which leads to less frequent predictions of positive examples than the class distribution would suggest.

HOMER shifts the trade-off between recall and precision to a more balanced level, increasing recall but losing precision. The reason is probably the smaller problems in terms of number of classes and consequently the increased density, which ascends to at least  $1/k = 10\%$ . This was already shown in the correlative behavior between number of partitions and precision in Figure 7.5. The effect can also be seen when using BR as multilabel base classifier technique for HOMER, but it is less pronounced since the plain BR itself produces more balanced results.

Due to the great differences in recall and precision between the algorithms, we decided to omit the Hamming losses in Table 7.3 though this measure is usually used for evaluating multilabel algorithms, since Hamming loss generally favors algorithms with high precision and low recall (cf. Section 2.7.3), which in this case means to favor CLR.

The F1 measure, which returns the harmonic mean between recall and precision, allows a more commensurate analysis in this particular case since it penalizes greater differences to a higher degree. Except on the *mediamill* dataset, for which the relatively high label density does not show a great impact on CLR’s recall, HOMER achieves the highest micro-averaged F1 value. In particular it outperforms BR on every dataset, which is especially interesting since HOMER is the direct competitor of BR in terms of computational costs. Similarly, HOMER+BR beats the plain BR except for *mediamill*, for which both algorithm are almost equal. HOMER+BR in general achieves less accurate predictions than using the pairwise approach as base classifier: in terms of F1 HOMER+BR is beaten on all datasets, in terms of recall and precision both algorithm are either almost equal (HOMER+BR slightly ahead) or HOMER+QCLR is clearly on top.

### 7.3.3.2 Computational Time

Basically the results of measuring the CPU time confirms our analysis in Section 7.2, as is demonstrated in the following. As shown in Table 7.3, HOMER is able to reduce the training time in comparison to plain BR approx. between 60% and 44% for using BR as base and between 33% and 10% for using QCLR. The first comparison is especially interesting since HOMER+BR has to train more base classifiers than BR: one classifier for each class at the leafs such as BR in addition to the classifiers in the inner nodes. However, this is done obviously with less training examples due to the filtering of examples at the inner nodes (cf. Section 7.2).

Comparing the two HOMER variants, we can observe that the overhead of training the pairwise classifiers (i.e. H+CLR – H+BR) is always less than training the one-against-all classifiers. Note that QCLR has to train the same classifiers as BR for the comparison to the calibrating artificial class plus the pairwise classifiers between relevant and irrelevant classes. This was anticipated in the formal analysis, however the reduction of the cardinality to the extreme case of 1 by appropriate clustering, for which pairwise decomposition slightly uses less training examples (approx.  $(k-1)/k$  less), does not justify the substantial margin. This can only be explicated with the super-linear training complexity of J48 with respect to the training size. The positive effect of smaller pairwise subproblems and super-linear learners was already analyzed for conventional flat CLR in Section 3.4.6.3 and is obviously also an argument for the combination with HOMER.

For testing, HOMER+BR is slightly slower than BR for the smaller *mediamill* and *jmlr2003*, but for the greater datasets *eccv2002* and *hifind* it requires less time. Although HOMER+BR has trained more base classifiers than the plain BR approach, it may invoke less base classifiers since great part of the classifier tree is pruned each time a meta-label is predicted as negative. This effect was already observed in the previous work on a dataset with almost 1000 classes (Tsoumakas et al. 2008). Note that prediction errors have a unfavorable impact on H-BR’s costs for problems with low label densities, since the false positives at the inner nodes increases, leading to further otherwise unnecessary evaluations. The formal analysis ignored this aspect, but the higher the label dimensionality, the closer the ratio becomes to the estimated asymptote.

HOMER+QCLR spends between 3% and 40% more time than BR, however, testing costs are so small for J48 compared to training time so that this increase is almost not noticeable. Again, the overhead for evaluating the additional pairwise classifiers in HOMER+QCLR only requires a small fraction of the time needed for the BR classifiers. Nevertheless, it is not possible to simply compute the overhead as difference between the time for HOMER+BR and +QCLR since prediction accuracy, especially precision, may also influence the classification time of QCLR (cf. Section 5.4.3). As expected, CLR requires the most computations for learning and predicting. However, the factor in training costs is proportional to the average label set size per example, which makes the costs acceptable for most of the multilabel problems since the label sets tend to be small. For prediction, the usage of QVoting is able to considerably reduce the costs in comparison to the evaluation of all pairwise base classifier while maintaining the advantage of the pairwise approach in terms of predictive performance.

In summary, the results on the testing time confirms our previous analysis. In particular we can observe that the several algorithms behave to each other as expected from Table 7.1. The example of the ratio between CLR and H-CLR demonstrates this: it ranges from 13 to 27, which is within the same order of magnitude as the estimated  $n/10$ .

## 7.4 Related Work and Discussion

HOMER is strongly related to existing approaches for learning hierarchical multilabel data, some of which were already discussed in the respective Section 2.5.6. The main difference is that those approaches assume an explicit hierarchical structure on the labels and are aimed at exploiting this. HOMER's objective in contrast is to reduce computational costs and increase effectivity. Consequently, this approach tries to preferably build up balanced hierarchies (in order to obtain beneficial balanced subproblems) and to maximally exploit label correlations (in order to reduce expensive branching and improve predictive accuracy). Moreover, using the real hierarchy potentially ignores dependencies between labels in different subtrees, which could be a further advantage of using an artificial but adapted and optimized structure.

Nonetheless, hierarchical multilabel classifiers often predict and are structured and trained the same way as in HOMER: test and training examples are passed from the root trough the tree according to label tests at the edges (in contrast to feature tests for decision trees for example). Cesa-Bianchi et al. (2006), Vens et al. (2008) and Zimek et al. (2010) e.g. adopt this approach or compare to a baseline working this way. However, the main difference is certainly that these approaches have labels at the inner nodes, which generally are also predicted if the path stops at these nodes, whereas HOMER uses metalabels and labels from  $\mathcal{L}$  can only be predicted at the leaves. Note that HOMER can easily be adapted in order to use a predetermined hierarchy on the labels, however, as explained above, HOMER follows a different objective which is probably not fully compatible to this approach.

A further difference to Zimek et al. is that they interpret the classifier outcomes as probabilities. Specifically, they approximate the conditional probability  $P(\lambda_u | \lambda_v \in P_x)$ ,



---

$\lambda_v \succ_{\tilde{h}} \lambda_u$  with  $h(\mathbf{x})$ . Hence the probability for a label  $\lambda_u$  becomes  $P(\lambda_u|\lambda_v) \cdot P(\lambda_v|\lambda_w) \cdots$  for  $\lambda_u \succ_{\tilde{h}} \lambda_v \succ_{\tilde{h}} \lambda_w \cdots$ . This is an interesting option for extending HOMER in order to additionally predict rankings on the predicted set of relevant and irrelevant labels whenever soft classifiers are used as base. Note that pairwise decomposition naturally predicts such scores in the form of number of votes while BR explicitly requires soft base classifiers.

## 7.5 Summary

This chapter provided an empirical and analytical study of the performance of HOMER. Compared to previous work (Tsoumakas et al. 2008), HOMER was extended by the integration of pairwise decomposition, particularly QVoting calibrated label ranking.

Interestingly, the results on four large multilabel datasets with a variety of characteristics showed that the instantiation of the multilabel learner of HOMER to QCLR can lead to better results compared to instantiating it to BR at a small expense in training and classification time. HOMER improves the training time of conventional BR and the difference is even more important for QCLR. In terms of classification time, HOMER substantially improves QCLR, while for BR the analytically proven benefits appear for the two largest datasets in terms of number of labels. Except for the *mediamill* dataset (where the differences are rather small), HOMER managed to improve the performance of the base multilabel learner (both BR and QCLR).

The main reason for employing HOMER was the scalability problem of QCLR in terms of memory with respect to the number of labels. The additional hierarchical decomposition and hence the pre-selection of considered pairwise preferences substantially reduces the amount of needed binary classifiers. In the same manner it provides a significant reduction in training and testing time for the pairwise CLR methods. It is also shown that HOMER is able to equilibrate recall and precision, especially for QCLR which is known to underestimate the number of labels per instance for problems with a low label density.



---

## 8 Exploitation of Label Dependencies in Parallel Tasks

In the previous chapters several improvements were presented in order to extend the application of the pairwise decomposition approach to domains with high dimensionalities. This enables us to consider the pairwise preference learning strategy to application areas which were unreachable before due to scalability constraints. One of these applications consists in the joint processing of several interconnected tasks, commonly referred to as *multi-task learning*. We consider in this chapter to simultaneously solve several multilabel tasks from different domains, which essentially results in a multiplication of the label dimensionality. However, this approach allows to exploit label dependencies across domains.

The starting point of this study is the following exemplary scenario: Books in a library are typically cataloged according to different types or domains of associated characteristics, e.g. genre, language, topic, epoch, author, etc. This type of annotation of objects is a very natural and common approach not only in the cataloging of texts (Section 8.4) but also e.g. when indexing music (Pachet and Roy 2009). Each of these mappings could be seen and treated as independent from each other. In reality, however, there may be dependencies between the different associated values from different domains. An author may write only in a specific language and focus exclusively on *crime fiction*. At the same time, crime fiction novels may often have *murder* as one of their topics, etc. Thus, if we consider to learn a model that automatically catalogs books in a library database as a text classification problem, for instance, it may be advantageous to consider all the parallel subproblems as a single large joint problem instead of tackling each subproblem separately.

In principle this is the same idea as in multi-task learning. In multi-task learning, we have a set of related learning problems (tasks), i.e. problems that have a common shared representation of their objects. It has been shown that learning these tasks simultaneously and jointly outperforms the common approach of learning them separately (single-task learning, cf. Section 8.1. The library example can be seen as a special multi-task learning scenario in which each categorization domain represents a separate task, and all tasks share the same representation of their objects (books have the same representation, e.g. the same bag of words, in every task).

Simultaneously, the approach of considering the whole task rather than each sub-task separately is in principle also the basic idea behind many multilabel classification algorithms. Instead of considering each label as a separate problem, as in the popular binary relevance (one-against-all) approach, most of the recent approaches try to imple-

---

itly or explicitly take into consideration existing label correlations in order to improve the predictive quality (cf. Section 8.1).

The approach that we propose is to consider the set of parallel multilabel tasks in the library as a single joint task, as in multi-task learning, and solve it with a conventional multilabel classification algorithm. Most of the recent and more sophisticated multilabel approaches may benefit from the parallel processing as they also benefit from the commonality in a conventional multilabel setting. We propose in this to our knowledge first work on the subject to use pairwise decomposition, which implicitly considers label relationships by learning preferences between pairs of labels. Furthermore, the advances presented in this thesis in handling many, even thousands of classes despite the quadratic number of models enable us to address the considerably increased complexity when the subtasks are joined.

## 8.1 Related Work

Approaches that try to explicitly exploit label dependencies include the early work of [McCallum \(1999\)](#), in which generative models for labelsets are generated as a mixture of topic based word distributions, the conditional random fields parameterized by label co-occurrences by [Ghamrawi and McCallum \(2005\)](#) and the label correlations conditioned maximum entropy method of [Zhu et al. \(2005\)](#), among others. A middle way is followed by [Dembczyński et al. \(2010a\)](#), [Read et al. \(2009\)](#) and their (probabilistic) classifier chains by stacking the underlying binary relevance classifiers with the predictions of the previous ones, and by [Cheng and Hüllermeier \(2009\)](#), whose k-NN approach stacks the appearances of labels in the neighborhood as new features. Indeed, adding label dependent features is a very popular approach in order to consider dependencies. Chapter 9 presents an approach relying on locally exceptional label constellations. Several other approaches are discussed within this context in Section 9.7.

However, the majority of the approaches implicitly consider dependencies by optimizing a loss on the predicted ranking of the labels. MMP perceptrons ([Crammer and Singer 2003](#)), Rank-SVM ([Elisseeff and Weston 2001](#)), Structural SVMs ([Tsochantaridis et al. 2005](#)) and the BP-MLL neural network algorithm ([Zhang and Zhou 2006](#)) e.g. rely on this. The latter approach is conceptually very similar to the multi-task neural networks of [Caruana \(1997\)](#), as both train a common network with several outputs denoting the labels, i.e. task outcomes. This is a popular approach in multi-task learning, also applied to Bayesian networks ([Bakker and Heskes 2003](#)). Other techniques try to develop special kernel functions which model inter-task relations ([Evgeniou et al. 2006](#)), or use statistical Dirichlet processes for the bayesian modeling ([Xue et al. 2007](#)).

A recent work in the field of natural language processing considers to jointly perform named entity recognition and syntactic parsing ([Finkel and Manning 2010](#)). The approach in Chapter 10 is similar, with the difference that we consider the recognition of each type of syntactic entity as a different task itself and merge these into one overall multilabel task.

A common problem to the referenced multilabel methods is their scalability in terms of number of labels, a factor which significantly increases when the subtasks are joined. Existing large scale approaches rely on the binary relevance decomposition (Montejo R  ez et al. 2004, Pouliquen et al. 2003, Tang et al. 2009) or using one-class classifiers (Vilalba and Cunningham 2009) (cf. also Section 6.5). However, solving each sub label relevance problem in a separate way would not change anything in comparison to solving it as multiple single-task problems in our proposed setting, neither computationally nor predictively.

## 8.2 Preliminaries

In multi-task learning, we assume several associated and interconnected tasks  $\mathcal{T}^{(i)}$ . The feature spaces  $\mathcal{X}^{(t)}$  of the different tasks  $t = 1 \dots k$  are supposed to be similar and share common features in multi-task learning. This is a precondition for the learning transfer: there has to be a link between the instances in task  $s$  and  $t$  for any link between the two output spaces to be recognized.

In this work we assume that all tasks share the same input space  $\mathcal{X} = \mathcal{X}^{(1)} = \dots = \mathcal{X}^{(k)}$  and that there is a common training set  $\mathbf{x}_1, \dots, \mathbf{x}_m \in \mathcal{T}_{rain}$  for all tasks  $t = 1 \dots k$ . This restriction corresponds to the common problem setting described in the introduction of Chapter 8. Each training example  $\mathbf{x}$  is hence associated with  $k$  outputs  $P^{(1)}, \dots, P^{(k)}$ , with  $P^{(t)} \subseteq \mathcal{L}^{(t)}$ ,  $t = 1 \dots k$  and  $\mathcal{L}^{(t)} = \{\lambda_1^{(t)}, \dots, \lambda_{n^{(t)}}^{(t)}\}$ . We define  $\mathcal{Y} = \mathcal{Y}^{(1)} \times \dots \times \mathcal{Y}^{(k)}$  correspondingly (cf. Section 2.4). We will denote this setting as *parallel tasks* in this work, however we will occasionally use *multi-task* as a synonym.

The learned multilabel classifier is a function  $h^{(t)} : \mathcal{X} \rightarrow 2^{\mathcal{L}^{(t)}}$ , or alternatively  $h^{(t)} : \mathcal{X} \rightarrow \mathcal{Y}^{(t)}$ , with  $\hat{P}^{(t)} = h^{(t)}(\mathbf{x})$  as the relevant labels predicted for test document  $\mathbf{x}$ . Multilabel classifiers commonly also predict a ranking  $\mathbf{r}^{(t)}$  on the labels, with  $r^{(t)}(\lambda^{(t)})$  returning the position of class  $\lambda^{(t)}$  in the relevance ranking (cf. Eq. 2.20).

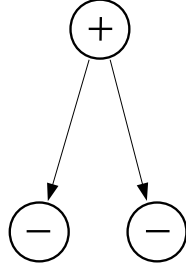
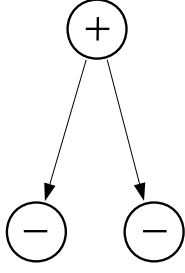
## 8.3 Parallel Task Learning

In order to benefit from the parallel alignment of the sub-task, the idea presented in this chapter is to simply join the different multilabel problems and treat them as a single large multilabel task. That means, we transform the *local* problems into one *global* problem with the training set  $\mathbf{x}_1, \dots, \mathbf{x}_m$  and the global training signals  $P_1^*, \dots, P_m^*$ , with  $P_i^* \subseteq$ ,  $i = 1 \dots m$ ,

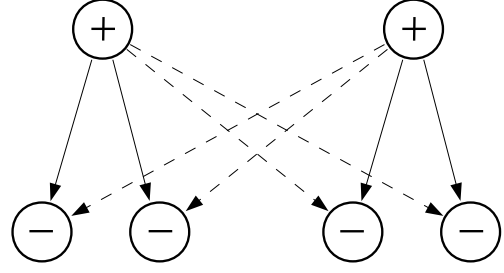
$$\mathcal{L}^* = \bigcup_{1 \leq t \leq k} \mathcal{L}^{(t)} \quad (8.1)$$

We define the  $\mathcal{L}^{(t)}$  as being disjoint, i.e.

$$\bigcup_{1 \leq t < s \leq k} \mathcal{L}^{(s)} \cap \mathcal{L}^{(t)} = \emptyset \quad n^* = |\mathcal{L}^*| = \sum_{1 \leq t \leq k} |\mathcal{L}^{(t)}| \quad (8.2)$$



**Figure 8.1:** Pairwise training on two separate tasks (left and right): circles with a plus/minus symbol represent the positive labels  $P$ /negative labels  $N$  a training example  $\mathbf{x}$  belongs to/does not belong to. The arrows represent the learned preferences.



**Figure 8.2:** Pairwise training on the global problem by joining the two tasks from Figure 8.1: the dotted arrows denote the additional learned relations for the current example  $\mathbf{x}$ .

After training the global multilabel learner, we obtain the global model  $h^* : \mathcal{X} \rightarrow \mathcal{L}^*$ ,  $\hat{P}^* = h^*(\mathbf{x})$ , which is then transformed back to the local classifiers

$$h^{(t)}(\mathbf{x}) = \hat{P}^{(t)} = \hat{P}^* \cap \mathcal{L}^{(t)} \quad (8.3)$$

The ranking function  $r^{(t)}(\lambda^{(t)}) := |\{\lambda_u^{(t)} \in \mathcal{L}^{(t)} \mid r^*(\lambda_u^{(t)}) \leq r^*(\lambda^{(t)})\}|$  is determined similarly.

As a convention, in the context of multilabel settings, we do not make any distinction in the notation of whether we are dealing with the global task or the subtasks and we will therefore omit the superscript.

### 8.3.1 Pairwise Classification for Parallel Tasks

In the pairwise decomposition method, one classifier is trained for each pair of classes. The trained classifiers are shown as arrows in the simple example in Figure 8.1.

The advantages from using pairwise binarization in contrast to the one-sided BR approach were already extensively discussed in Section 3.5. Furthermore, we expect to further benefit from the following characteristics specific to the parallel tasks setting:

Firstly, it was observed in Chapter 4 that one additionally introduced virtual label and hence the additional learned preferences could already slightly improve the predictions. We expect to benefit from this effect for the many additional connections when joining to a global model. Figure 8.2 shows this on an example of two small parallel tasks. Assuming  $k$  parallel tasks of equal size  $n$ , the number of base classifiers increases by a factor of  $\frac{kn(kn-1) \cdot 2}{k \cdot n(n-1) \cdot 2} = \mathcal{O}(k)$ . From the point of view of scalability, this increase is manageable. On the other hand, it means a considerably increase in the preference information available for inducing more accurate model. It also provides additional exploitable information to the next aspect.

Secondly, pairwise classification implicitly exploits label dependencies since the models are specifically trained to detect exclusion of labels. Remember that a base classifier  $h_{u,v}$  is trained exactly with all the examples for which  $\lambda_u$  and  $\lambda_v$  are mutually exclusive. A positive prediction of  $h_{u,v}$  could hence be interpreted as the implication  $\lambda_u \in P \rightarrow \neg(\lambda_v \in P)$  holding on the current test instance. Since currently the base learners are not supposed to model something different than exclusion implications, we have to be cautious with this interpretation and therefore the estimations are currently just counted as simple votes and aggregated into a ranking. We plan to extend our approach in order to support extended expressiveness. In addition, incorporating (a priori) label constraints by incorporating them into the training process and by correcting predictions is being studied in ongoing work (Park and Fürnkranz 2008). In Chapter 9 we incorporate additional features into the instance representation that encode local label dependencies.

### 8.3.2 Calibration

To convert the resulting ranking of labels into a multilabel prediction, we use the calibrated label ranking approach (cf. Section 3.4.5). Figure 8.3 shows the introduction of the additional calibrating label for the two task in Figure 8.1. As already known, the resulting  $n$  additional binary classifiers  $\{h_{u,0} \mid u = 1 \dots n\}$  are identical to the classifiers that are trained by the binary relevance approach. This holds also for the parallel task setting, as can be seen in Fig. 8.4 and 8.5, making the approach also easily applicable to this setting.<sup>60</sup>

Since we are generally also interested in the ranking performance in our evaluation, we rely on the full CLR rather than the aggregation optimizing QVoting extension (Chapter 5).

For the text data, we use CLR together with the simple but fast perceptrons as base learners, leading to the (incrementally trainable) Multilabel Pairwise Perceptrons algorithm and its dual variant *DMLPP* for large numbers of labels, as this combination has shown to be efficient as well as effective in multiple occasions in this work.

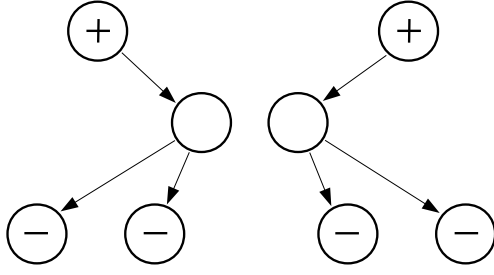
### 8.3.3 HOMER with QCLR

Since the pairwise subproblems in one of the datasets in Section 8.4 are not linearly separable, we are not able to use the efficient pairwise perceptron approach.<sup>61</sup> And due to the high number of labels and the higher complexity of the common non-linear classifiers (both time and memory), plain CLR with a different base learner is also not viable so far.

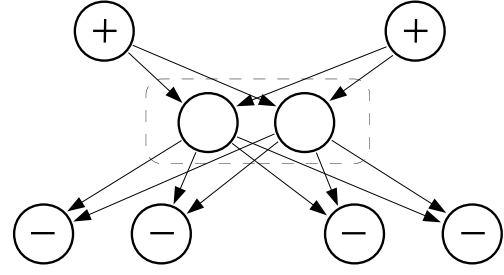
We elude this problem by using the previously introduced *HOMER* algorithm as the meta-learner for the CLR approach, which was specifically developed in order to handle

<sup>60</sup> This approach may disadvantage smaller sub-tasks, however we evaluate mainly independently from the right thresholding and thus leave the analysis for future work.

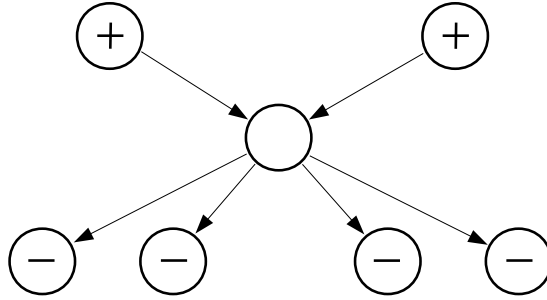
<sup>61</sup> Unfortunately, the current implementation of *DMLPP* does not support kernels. We hope to add this soon.



**Figure 8.3:** The two additional virtual labels  $\lambda_0^{(s)}$  and  $\lambda_0^{(t)}$  and additional preferences trained for the separate calibrations of the two tasks from Figure 8.1. During prediction, the ranking of the obtained votes per label is split at the position of the virtual label.



**Figure 8.4:** Calibration in the global task: both virtual labels  $\lambda_0^{(s)}$  and  $\lambda_0^{(t)}$  are always located at the same level between positive and negative labels  $P^*$  and  $N^*$ , therefore the base classifiers to and from the two virtual classes are trained with the same examples.



**Figure 8.5:** Calibration in the global task: the virtual labels  $\lambda_0^{(s)}$  and  $\lambda_0^{(t)}$  were merged to one unique calibration label  $\lambda_0^*$ . In all three cases in Figures 8.3, 8.4 and 8.5 the same classifiers discriminating between the virtual labels and the remaining labels are learned.

multilabel problems with a large number of labels (cf. Chapter 7). For the decision, which labels to join to one metalabel at the inner nodes, we employ the balanced k-means algorithm which preferably aggregates labels that are correlated.

At first sight breaking up the problem into smaller ones may sound contradictory to our proposed approach of considering several subtasks as a unique multilabel task. The reason for following the subdivision approach nevertheless is that we expect that we are still able to benefit from the additional possible inter-task label correlations, since the label clustering method in HOMER creates subproblems preserving as much label correlation information as possible.

In addition to this retained advantage, we saw that using HOMER also helped in order to balance recall and the high precision usually produced by the conservative plain CLR.



**Table 8.1:** Statistics for *EUR-lex* and *rcv1*. Label *density* indicates the average number of labels per instance  $\bar{d}$  relative to the total number of classes  $n$ ,  $m$  denotes the number of documents,  $a$  the number of used features.

dataset	$n$	$m$	$a$	$\bar{d}$	density
<i>EUR-lex</i>	4567	19348	5000	8.82	0.19 %
<i>sm</i>	201	"	"	2.21	1.11 %
<i>dc</i>	410	"	"	1.29	0.31 %
<i>ev</i>	3956	"	"	5.31	0.13 %
<i>rcv1</i>	103	804414	25000	3.24	3.15 %
<i>ccat</i>	34	"	"	1.44	4.24 %
<i>ecat</i>	26	"	"	0.41	1.58 %
<i>gcat</i>	33	"	"	0.70	2.12 %
<i>mcat</i>	10	"	"	0.69	6.90 %

## 8.4 Datasets

We used the *EUR-Lex* dataset (cf. Section 6.1) for evaluation since it naturally provides documents with categorization from different domains. The three different classification schemes subject matter (*sm*) with 201 classes, *directory code* (*dc*) with 410 classes and EUROVOC (*ev*) with 3956 classes constitute the local domains, while the global task considers 4567 labels plus the virtual class. The same preprocessing and cross validation distribution as in Section 6.1 was used. The dataset was processed with DMLPP trained over two epochs.

The *hifind* dataset contains 32,769 music titles annotated with 632 different labels (Pachet and Roy 2009). The labels specify (mainly acoustic) characteristics of the categorized songs which can be divided into 17 distinct domains. Some of the subtasks were intended to be single-label (binary or multiclass), however for all of them the number of distinct labelsets is greater than the number of classes. Following Tsoumakas et al. (2008) and Chapter 7, we trained HOMER with a cluster size of 7 (if possible) in combination with QCLR and the J48 implementation of C4.5 (Witten and Frank 2005) as base classifiers on the first 16,452 examples and tested on the remaining 16,519.

As can be seen from the descriptions and Tables 8.1 and 8.2, the previous two real-world datasets fit perfectly to the illustrated library example in Section 8 and are hence prototypical for our parallel tasks setting. In addition, we simulated a multi-task setting on the Reuters datasets, for which we also recognized parallel dataset characteristics, although the corpus is normally only seen from the multilabel point of view.

We applied MLPP on *rcv1* with the same setup as in Section 4.6.1. The 103 categories of the dataset are organized in a hierarchy with four main sub nodes: government/social (*gcat*), markets (*mcat*), economics (*ecat*) and corporate/industrial (*ccat*). We chose these four subsets as the domains of the tasks in the multi-task setting, although the classes

**Table 8.2:** Statistics for *hifind*. The same notation is used as in Table 8.1.

dataset	$n$	$m$	$a$	$d$	density
<i>hifind</i>	623	32971	98	37.3	5.98 %
<i>character</i>	37	"	"	3.97	10.7 %
<i>country</i>	27	"	"	0.98	3.64 %
<i>dynamics</i>	4	"	"	0.99	24.8 %
<i>epoch</i>	16	"	"	1.03	6.42 %
<i>genre</i>	31	"	"	2.65	8.53 %
<i>instruments</i>	100	"	"	5.09	5.09 %
<i>language</i>	16	"	"	1.01	6.30 %
<i>metric</i>	10	"	"	1.00	9.96 %
<i>mood</i>	59	"	"	5.27	8.94 %
<i>period</i>	2	"	"	0.004	0.24 %
<i>popularity</i>	3	"	"	0.97	32.5 %
<i>rhythmics</i>	10	"	"	1.17	11.7 %
<i>setup</i>	25	"	"	2.25	8.98 %
<i>situation</i>	74	"	"	5.26	7.11 %
<i>style</i>	158	"	"	1.21	0.77 %
<i>tempo</i>	8	"	"	0.99	12.4 %
<i>variant</i>	43	"	"	3.46	8.04 %

therein are actually from the same type (topic categories) and it is therefore justified to treat them jointly from the beginning. However, a common binary benchmark dataset is based on this subdivision (Chang and Lin 2012). For the future we plan to use the additional associations contained in the corpus to 365 industry categories and 366 region categories, which have hardly received any attention yet in the literature.

## 8.5 Evaluation

Table 8.3 shows the results on the *EUR-Lex* tasks in terms of recall, precision, subset accuracy and the ranking losses *RANKLOSS*, average precision and idealistic F1 (cf. Section 2.7). We compare the locally trained classifiers to the globally trained models, which were afterwards reduced again to local predictors (cf. Section 8.3).

The first appreciable observation is that our parallel task (PT) approach considerably decreases recall and gains precision. This is due to the effect that calibration leads to cautious predictions when the number of labels is high and hence the label density is low (cf. Section 4.6.9). For the ranking based losses, the PT approach sometimes only slightly but always significantly outperforms the conventional method.<sup>62</sup> The subset accuracy is

<sup>62</sup> Wilcoxon signed-rank test,  $\alpha = 5\%$ , cf. Section 2.10.2.

**Table 8.3:** Results of DCMLPP on the *EUR-Lex* dataset. First row: results on the global dataset. Next blocks: results trained on sub-tasks in first rows, respectively. Trained on all parallel tasks (PT) in second rows. Last block: mean on all sub-tasks. Last row: number of wins of PT model over local model. Bold entries show the winner, italics a significant difference on the cross validation fold results.

	REC	PREC	ACC	RANKLOSS	AVGP	F1 <sub> p </sub>
<i>EURlex</i>	36.64	76.48	0.284	1.683	63.20	58.34
<i>sm</i>	<b>64.50</b>	75.48	33.29	0.874	83.26	75.25
PT	57.34	<b>85.36</b>	<b>36.42</b>	<b>0.851</b>	<b>84.45</b>	<b>76.96</b>
<i>dc</i>	<b>54.23</b>	77.11	<b>45.95</b>	0.844	81.05	71.42
PT	48.09	<b>83.94</b>	44.98	<b>0.840</b>	<b>82.20</b>	<b>73.25</b>
<i>ev</i>	<b>25.48</b>	66.63	<b>0.636</b>	2.325	53.35	48.59
PT	25.22	<b>67.10</b>	0.610	<b>2.307</b>	<b>53.47</b>	<b>48.71</b>
<i>mean</i>	<b>48.07</b>	73.07	26.63	1.348	72.55	65.09
PT	43.55	<b>78.80</b>	<b>27.34</b>	<b>1.333</b>	<b>73.37</b>	<b>66.31</b>
wins	0	<b>3</b>	1	<b>3</b>	<b>3</b>	<b>3</b>

again influenced by the conservative estimation of the calibration. A look at the *IsERR* error rates demonstrates this: while the global model loses for *ds* and *ev*, *IsERR* indicates that 69.94% are achievable by PT in contrast to 67.07% on *ds* and 3.45% instead of 3.37% on *ev* (cf. Section 2.7.4).

Note also that we have to be very cautious when comparing the task-averaged measures in the last blocks since the tasks are indeed parallel, but the measures are nevertheless computed on different label domains. For this reason the non-parametric Wilcoxon signed-rank test is used were applicable.

Table 8.4 shows the averages on the 16 tasks of the *hifind* dataset (task *period* was omitted since no classifier could be locally learned). We can observe the opposite behavior with respect to recall and precision using HOMER. These differences between recall and precision are more pronounced on the smaller tasks, which indicates that this might be related to the smaller proportion of number of labels to cluster size, since the smaller this proportion the greater precision and the smaller the recall in Section 7.3.2.3. Unfortunately, it is not possible to retrieve ranking losses for HOMER. Nevertheless, the gain in *REC* for the globally trained model outweighs the loss in *PREC* in terms of the less specific *F1* and *ACC*.

And more interestingly, this shows that it might be beneficial to join the parallel tasks although the base learner again breaks down the global task into smaller independent problems. For HOMER, this is probably due to the effective clustering of the generated subproblems so the information contained in the label correlations are preserved as much as possible. Reversely, this demonstrates the effectiveness of our parallel task setting since it shows the degree of additional information contained in the inter-domain correlations

**Table 8.4:** Results of HOMER on the *hifind* dataset, only the means and number of wins are shown. Italic values indicate a statistically significant difference between the means on the 16 sub-tasks. Measures based on rankings are omitted since the used base classifier only predicts labelsets.

	REC	PREC	ACC	F1
<i>hifind</i>	56.51	51.95	0.012	54.13
mean	50.96	<b>53.72</b>	23.11	51.77
PT	<b>56.33</b>	51.82	<b>24.32</b>	<b>53.74</b>
wins	<b>16</b>	4	7	<b>13</b>

**Table 8.5:** Results on the *rcv1* dataset of MLPP. Each block shows the direct comparison between locally and globally learned model, as in Table 8.3.

	REC	PREC	ACC	RANKLOSS	AvGP	F1 <sub> P </sub>
<i>rcv1</i>	80.32	83.89	49.78	0.526	93.35	87.22
<i>ccat</i>	<b>81.17</b>	76.27	66.16	0.549	97.37	88.71
PT	79.81	<b>80.85</b>	<b>69.54</b>	<b>0.528</b>	<b>97.42</b>	<b>89.00</b>
<i>ecat</i>	<b>70.09</b>	71.17	87.12	0.117	99.49	91.95
PT	68.90	<b>79.02</b>	<b>89.37</b>	<b>0.109</b>	<b>99.49</b>	<b>92.43</b>
<i>gcat</i>	<b>79.18</b>	81.67	83.57	0.158	99.10	91.36
PT	78.74	<b>84.79</b>	<b>85.07</b>	<b>0.149</b>	<b>99.13</b>	<b>91.78</b>
<i>mcat</i>	<b>89.12</b>	87.61	89.72	0.125	99.79	98.20
PT	88.39	<b>90.96</b>	<b>90.94</b>	<b>0.114</b>	<b>99.80</b>	<b>98.36</b>
mean	<b>79.89</b>	79.18	81.64	0.237	98.94	92.55
PT	78.96	<b>83.91</b>	<b>83.73</b>	<b>0.225</b>	<b>98.96</b>	<b>92.89</b>
wins	0	4	4	4	4	4

and that it can be effectively exploited. However, it would be interesting in this context to analyze the performance if the reverse way is followed, i.e. training on the local task and then aggregating it to a prediction for the global task. We leave this for future work.

The same conclusion is drawn from the results on the Reuters dataset in Table 8.5. Again, as on *EUR-Lex*, we can see the preference for high precision and lower recall of the global approach. However, the improvement on the remaining measures is clearer, even on the subset accuracy, though the differences in the nearly perfect AvGP results are almost not perceptible.

---

### 8.5.1 Additional Experiments

We ran also experiments with MMP on the text datasets, the only other algorithm known to the author at the time of writing that is able to handle such combinations of great number of instances, features and labels and that takes label correlations into account. Recall that using BR results in the semantically same one-against-all classifiers, globally as well as locally. Unfortunately, MMP has shown to be very susceptible to overfitting (cf. Section 4.6.6) and as expected we obtained generally worse results for the training on the global task.

Furthermore, we tried to learn on the *rcv1* data with the popular LibLINEAR algorithm as base learner (Fan et al. 2008), which is known to be efficient and effective on text data. Unfortunately, the training on the global task had not completed after two days using our framework. However, the results of LibLINEAR on the local task showed that MLPP trained globally is very comparable despite using plain, non-optimized perceptrons and only one pass over the training data. Requiring only 12 minutes, MLPP was also much faster than LibLINEAR on the smaller problems.

## 8.6 Summary

The starting point of this study was the recognition of a common characteristic of many real world problem, namely the mapping of the same object to concepts from several different domains. We referred to such problems as *parallel tasks* and evaluated the straightforward approach of joining the subtasks to one large global multilabel problem. As in the more general multi-task learning setting, we expected to benefit from the additional information obtained through the aggregation of the labelsets.

We showed that multilabel algorithms which consider label correlations are able to effectively exploit the label correlations. In particular, the highly scalable and efficient pairwise perceptrons algorithms improved the quality of the predicted rankings. Perhaps more surprising and pleasing was the insight that HOMER allows also less scalable base learners to take advantage of the parallel task setting, though the used mechanism is to divide the original problem into smaller subproblems, which is in a certain sense directly opposed but actually compatible to the proposed approach.

With the (dual) pairwise perceptrons ensembles, QVoting enhanced calibration, and HOMER we have created a frame in which it is potentially possible to deal with any of the challenges of pairwise multilabel classification outlined in Chapter 1, even if they appear combined. The work presented in this chapter demonstrates a first application beyond pure multilabel classification for which this framework provided the appropriate instruments.

This first evaluation of parallel tasks in the multilabel setting leaves several possibilities for future work. The more explicit exploitation of label correlations in pairwise decomposition is an ongoing issue (cf. Section 8.3.1). Furthermore, different label correlation respecting algorithms could be compared. Actually, this setting could effectively be used in practice in order to analyze to which degree a multilabel algorithm takes label corre-

---

lations into account. This particular property of multilabel algorithm makes it interesting to try to apply them on the more general multi-task learning setting, in which the objects in the tasks are not longer parallel but only similar. Of course, the opposite approach of incorporating ideas and mechanisms from multi-task learning is also very interesting.

---

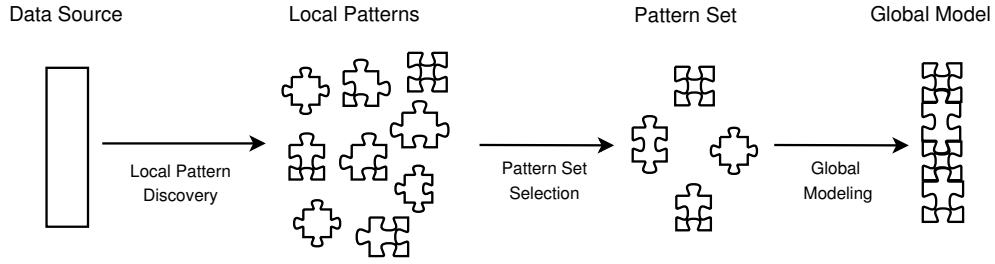
## 9 High-Order Dependencies with Patterns from Subgroup Discovery

Decompositive approaches such as binary relevance ignores interdependencies between labels, so that it can be expected that the predictive performance improves if label dependencies are taken into account. When, for instance, one considers a dataset where each label details the presence or absence of one kind of species in a certain region, the existing food chains between the species cause a plethora of strong correlations between labels. But interplay between species is more subtle than just correlations between pairs of species. It has, for instance, been shown (Paine 1966) that a food chain between two species (the sponge *Haliclona* and the nudibranch *Anisodoris*) may be displaced depending on the presence or absence of a third species (the starfish *Pisaster ochraceus*), which is not directly related to the species in the food chain. Apparently, there is some conditional dependence relation between these three species. The ability to consider such interplay is an essential element of good multilabel classifiers, as was already pointed out in Chapter 8.

In that chapter we presented empirical evidence for the exploitation of label dependencies by the pairwise decomposition approach. However, this framework is only able to detect certain types of dependencies and only dependencies of second order, i.e. between pairs of labels. The approach presented in this chapter encodes label dependencies of second and higher order directly in the feature space, so that the specific limitations of the learner in use can drop. For the pairwise decomposition this means to enhance the ability to consider label correlations and the following study investigates this aspect.

Hence, we propose to incorporate locally exceptional interactions between labels in multilabel classification, as an instance of the LeGo framework (Fürnkranz and Knobbe 2010, Knobbe et al. 2008). In this framework, the knowledge discovery process is split up in several steps: first local models are found, each representing only part of the data, then a sensible subset of these models is selected, and finally this subset is employed in the construction of a global model. The main idea here is that we can use straightforward classification methods for building a global classifier, if the local exceptionalities in interactions between labels are represented by features constructed from patterns found in the local modeling phase.

We propose to find patterns representing these locally exceptional interaction through an instance of Exceptional Model Mining (Leeuwen 2010, Leman et al. 2008); a framework that can be seen as an extension of traditional Subgroup Discovery. The particular instance we consider (Duivesteijn et al. 2010) models the conditional dependencies between the labels by a Bayesian network, and strives to find patterns for which the learned



**Figure 9.1:** The LeGo framework (from Frnkranz et al. 2012).

network has a substantially different structure than the network learned on the whole dataset.

These patterns can each be represented by a binary feature of the data, and a contribution of this chapter is the demonstration that the integration of these features into the classification process improves classifier performance. On the other hand, we expect the newly generated binary features to be expressive enough in order to being able to replace the original features. We also investigate this aspect and particularly the impact on efficiency.

## 9.1 Preliminaries

In this section, we briefly recall the main concepts upon which our work builds, namely the LeGo framework for learning global models from local patterns (Section 9.1.1) and multilabel classification (Section 9.1.2). We will then conclude with the problem formulation (Section 9.1.3).

### 9.1.1 The LeGo Framework

As mentioned, the work in this paper relies heavily on the LeGo framework (Frnkranz and Knobbe 2010, Knobbe et al. 2008). This framework assumes that the induction process is not executed by running a single learning algorithm, but rather consists of a number of consecutive steps. Specifically, in the first step, a local pattern discovery algorithm is employed in order to obtain a number of informative patterns, which can serve as relevant features to be used in the subsequent steps. In the second and third step, the patterns are filtered with the aim of reducing redundancy, and the selected patterns are combined in a final global model, which is the outcome of the process. The central idea of the LeGo framework is that, instead of attacking the induction task in the original representation, we transform it by an automated process to a representation that already resolves a number of complexities in the original task, after which the alternative representation can be easily solved with a standard global modeling technique. For this automated process,



---

any of the existing pattern mining algorithms can be employed, thus benefiting from the wealth of pattern mining algorithms that has grown over the last decade.

So why would one use a LeGo approach, with the associated increased computational cost, instead of applying a regular, single-step, algorithm? The main reason for doing so is the expected increase in accuracy of the final model, as a result of the higher level of exploration that is involved in the initial local pattern discovery step. Typically, standard global modeling techniques employ some form of greedy search, and in complex tasks, subtle interactions between attributes may be overlooked as a result of this. In most pattern mining methods however, extensive or even exhaustive consideration of combinations of attributes is quite common. When employing such exploratory algorithms as a form of preprocessing, one can think of the result (the patterns) as partial solutions to local complexities in the data. The local patterns, which can be interpreted as new virtual features, still need to be combined into a global model, but potentially hard aspects of the original representation will have been accounted for already. As a result, relatively straightforward methods, such as Support Vector Machines with linear kernels, can be used in the global modeling step.

The LeGo approach has shown its value in a range of settings ([Fürnkranz and Knobbe 2010](#)), particularly regular binary classification ([Knobbe and Valkonet 2009](#), [Sulzmann and Fürnkranz 2008](#)), but we have specific reasons for choosing this approach in the context of multilabel classification. It is often mentioned that in MLC, one needs to take into consideration potential interactions between the labels, and that simultaneous classification of the labels may benefit from knowledge about such interactions (cf. [Section 2.6, 9.7](#)).

In a recent publication ([Duivesteijn et al. 2010](#)), an algorithm is outlined that finds local interactions amongst multiple targets (labels) by means of an extended Subgroup Discovery (SD) method. The method is an instance of the Exceptional Model Mining (EMM) framework ([Leeuwen 2010](#), [Leman et al. 2008](#)), which suggests a discovery approach involving multiple targets, and using local modeling over the targets in order to find subsets of the dataset where unusual (joint) distributions over the targets can be observed. In the work of [Duivesteijn et al. \(2010\)](#), one instance of EMM was presented that deals with discrete targets, and employs Bayesian Networks in order to find patterns corresponding to unusual distributions and thus interesting interactions between targets. This Bayesian instance of EMM ideally fits the need in MLC for explicit representations of local and unusual combinations of labels.

### 9.1.2 Multilabel Classification

In order to evaluate our approach and the impact of the additional information from the local patterns on multilabel learner, we have made a selection of approaches that consider label dependencies to different degrees.

The widely used binary relevance approach tackles a multilabel problem by learning a separate classifier for each label (cf. [Section 3.1](#)). Obviously, BR ignores possible interdependencies between classes since it learns the relevance of each class independently.

Attributes	Labels $\in \{0, 1\}^n$
$x_{1,1}, \dots, x_{1,a}$	$y_{1,1}, \dots, y_{1,n}$
$\vdots \quad \ddots \quad \vdots$	$\vdots \quad \ddots \quad \vdots$

(a) input training set

Attributes	Class $\in \mathcal{L}$
$x_{1,1}, \dots, x_{1,a}$	$\lambda_{i_1^1}$
$\vdots \quad \ddots \quad \vdots$	$\vdots$
$x_{1,1}, \dots, x_{1,a}$	$\lambda_{i_{ P_1 }^1}$

(b) *multiclass* (MC) decomposition  
(for feature selection)

Attributes	Class <sub>1</sub> $\in \{0, 1\}$	...	Attributes	Class <sub>n</sub> $\in \{0, 1\}$
$x_{1,1}, \dots, x_{1,a}$	$y_{1,1}$		$x_{1,1}, \dots, x_{1,a}$	$y_{1,n}$

(c) *binary relevance* (BR) decomposition

Attributes	Class $\in 2^{\mathcal{L}}$
$x_{1,1}, \dots, x_{1,a}$	$P_1$

(d) *label powerset* (LP) transformation

**Figure 9.2:** Decomposition of multilabel training sets into binary (BR) or multiclass problems (MC, LP).  $P_j = \{\lambda_{i_k^j} \mid y_{j,i_k^j} = 1, k = 1, \dots, |P_j|\}$  denotes the  $|P_j|$  positive classes of example  $x_j$ . MC replicates each instance  $|P_j|$  times, resulting in a multiclass problem. In LP the (single) target value of an instance  $x_i$  is from the set  $\{P^i \mid i = 1 \dots m\} \subseteq 2^{\mathcal{L}}$  of the different label subsets seen in the training data.

One way of addressing this problem is by using *classifier chains* (CC) (Read et al. 2011), which are able to model label dependencies since they stack the outputs of the models: the prediction of the model for label  $\lambda_i$  depends on the predictions for labels  $\lambda_1, \dots, \lambda_{i-1}$  (see also the discussion in Section 3.5.7).

An alternative approach is the pairwise decomposition in calibrated label ranking (cf. Section 3.4). CLR learns binary classifiers  $h_{u,v}(\mathbf{x}) \rightarrow (\lambda_u \succ_{\mathbf{x}} \lambda_v) \vee (\lambda_v \succ_{\mathbf{x}} \lambda_u)$ , which predict for each label pair  $(\lambda_u, \lambda_v)$  whether  $\lambda_u$  is more likely to be relevant than  $\lambda_v$ . Thus, CLR (implicitly) takes correlations between pairs of labels into account (cf. Section 8.3.1).

Finally, a simple way to take label dependencies into account is the *label powerset* (LP) approach (cf. Section 3.2), which treats each possible combination of labels that occurs in the training data as a separate label of a classification problem.

The functioning of BR and LP is reviewed in Figure 9.2 in the context of feature subset selection (cf. Section 9.3).

---

### 9.1.3 Problem Statement

The main question this paper addresses is whether a LeGo approach can improve multi-label classification, in comparison to existing classification methods that do not employ a preliminary local pattern mining phase. Thus, our approach encompasses:

1. find a set  $\mathcal{P}$  of patterns representing local anomalies in conditional dependence relations between labels, using the method introduced by Duivesteijn et al. (2010),
2. filter out a meaningful subset  $\mathcal{S} \subseteq \mathcal{P}$ ,
3. use the patterns in  $\mathcal{S}$  as constructed features to enhance multilabel classification methods.<sup>63</sup>

The following sections will explore what we do in each of these steps.

## 9.2 Local Pattern Discovery Phase

In order to find the local patterns, with which we will enhance the feature set used in the multilabel classifiers, we employ an instance of Exceptional Model Mining (EMM). This instance is specifically tailored to find subgroups in the data where the conditional dependence relations between a designated set of target features (our labels) is significantly different from those relations on the whole dataset. Before we recall the EMM instance of our choice in more detail, we will first outline the general EMM framework.

### 9.2.1 Exceptional Model Mining

Exceptional Model Mining is a framework that can be considered an extension of the traditional Subgroup Discovery (SD) framework, a supervised learning task which strives to find *patterns* (defined on the input variables) that satisfy a number of user-defined *constraints*.

A pattern is a function  $p : \mathcal{X} \rightarrow \{0, 1\}$ , which is said to *cover* a data point  $\mathbf{x}_i$  if and only if  $p(x_{i,1}, \dots, x_{i,a}) = 1$ . Even though a pattern represents knowledge about a local complexity in the data, we do not need to include this complexity in the definition of the concept.

The user-defined constraints typically include lower bounds on the number of data points the pattern covers and on the quality of the pattern, which is usually defined on the output variables (but the definition may also incorporate input variables). One could also impose additional constraints such as an upper bound on the complexity of the pattern. A run of an SD algorithm results in a quality-ranked list of patterns satisfying the user-defined constraints.

---

<sup>63</sup> Pay attention to the slight overload of the letter  $p$  for the positive labelset  $P$  (upper case), the pattern set  $\mathcal{P}$  (calligraphic) and the pattern or pattern function  $p$  (lower case).

---

In traditional SD, we have only one single target variable. Most commonly used pattern quality measures contain a component indicating how the distribution of the target variable in the data points covered by the pattern differs from its distribution in the whole dataset. Since a large deviation is easily achieved in a small subset of the data, a typical quality measure also contains a component indicating the number of data points covered by the pattern.

EMM extends SD by allowing for more complex target concepts. It assumes a partition of all features in the dataset into two sets: the first subset is used to generate the candidate patterns, the second subset is used for evaluating the candidate patterns. On the second feature subset a model class is defined, and an exceptionality measure  $\varphi$  for that model class is selected. Such a measure assigns a quality value  $\varphi(p)$  to a candidate pattern  $p$ . EMM algorithms traverse a search lattice of candidate patterns, constructed on the first feature subset, in order to find patterns that have exceptional values of  $\varphi$  on the second feature subset.

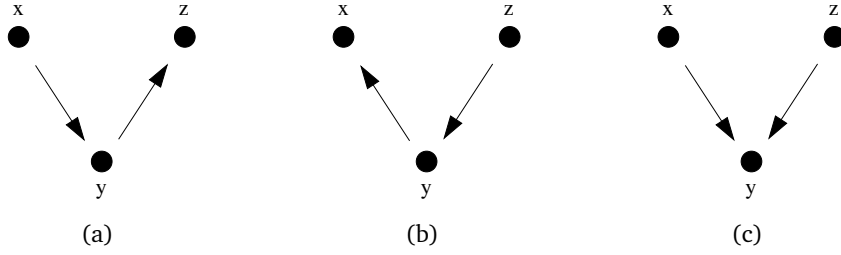
As an example, consider an EMM instance where two features are selected as the second subset. Then the model class could be correlation models between these features, and the exceptionality measure could be the correlation coefficient between them. This instance of EMM searches for patterns in the data for which the two designated features have an exceptionally high positive correlation. If one is interested in both positive and negative correlations, one could use the absolute value of the correlation coefficient as exceptionality measure. If one is interested in an exceptional ratio between the two designated features, one could take regression models as the model class, and then define an appropriate exceptionality measure based on for example the slope of the regression line.

When the feature space is not small enough to allow exhaustive exploration, the search lattice is usually traversed with a beam search strategy, rendering the search level-wise. The search starts with a candidate pattern covering the entire dataset. On each level the current candidate pattern set is refined, creating many new candidates from each old candidate. Each new candidate is formed from an old candidate by imposing one additional restriction on one feature from the first feature subset. Then, the best  $w$  patterns according to the quality measure at hand are selected as the new candidate pattern set for the next level.

The beam search strategy finds a middle ground between a hill-climbing search and exhaustive exploration: selecting the  $w$  best patterns at each level makes the search process tractable and keeps it focused, while considering  $w$  alternatives on each level makes the process less likely to end up in a local optimum. The search can be further constrained by setting an upper bound on the search level, and a lower bound on the coverage of the candidate patterns.

### 9.2.2 Exceptional Model Mining meets Bayesian Networks

As discussed in Section 9.1, we assume that the features in our dataset are partitioned in the set of  $a$  attributes, that can be from any domain, and the set of  $n$  labels, that are



**Figure 9.3:** Example Bayesian networks (based on Duivesteijn et al. 2010, Fig. 1).

assumed to be discrete. The EMM instance we employ (Duijvesteijn et al. 2010) proposes to use Bayesian networks over those  $n$  labels as model class. These networks are directed acyclic graphs (DAGs) that model the conditional dependence relations between their nodes. A pattern has a model that is exceptional in this setting, when the conditional dependence relations between the  $n$  labels are significantly different on the data covered by the pattern than on the whole dataset. Hence the exceptionality measure needs to measure this difference. To this end, first a Bayesian network on the labels is learned for the whole dataset. Then, for each candidate pattern under consideration, a Bayesian network on the labels is learned from only the records in the dataset covered by the pattern.

The exceptionality measure contains a component indicating the extent to which the Bayesian network for the candidate pattern is different in structure from the Bayesian network for the whole dataset. Because of the peculiarities of Bayesian networks, we cannot simply use traditional edit distance between graphs (Shapiro and Haralick 1985) here. Instead, a variant of edit distance for Bayesian networks was introduced, that basically counts the number of violations of the famous theorem by Verma and Pearl on the conditions for equivalence of Bayesian networks (Verma and Pearl 1990).

Essentially, we count the distinct edges between two graphs ignoring their orientation. Hence, the distance between the networks (a) and (b) in Figure 9.3 is zero. However, we preprocess the graphs so that *unmarried parents* are connected, i.e. nodes implying (pointing to) the same child node. Thus, the distance between (a) or (b), respectively, and (c) is  $1/3$ . Suppose (a) is the network learned on the whole dataset, then we would prefer the pattern corresponding to (c) since it is more different and hence more exceptional than (a).

Notice that the EMM algorithm takes quite some time to complete: for each candidate pattern a Bayesian network is built over the labels, which can be done in  $\mathcal{O}(n^{2.376})$  time (Duijvesteijn et al. 2010). Since many candidate patterns are considered, this is a high price to pay, even with a relatively small number of labels. However, as stated in Section 9.1, in the LeGo approach the patterns resulting from the EMM algorithm can be considered partial solutions to local complexities in the data. These solutions do not need to be recomputed every time a classifier is built. Hence, the EMM algorithm needs to be executed only once, and we can afford to invest quite some computation time for

this single run. Additionally, the investment needed to compare Bayesian networks as opposed to, for instance, comparing pairwise dependencies between labels, allows us to find anomalies in non-trivial interplay between variables.

After running the EMM algorithm, we obtain a set  $\mathcal{P}$  of patterns each representing a local exceptionality in the conditional dependence relations between the  $n$  labels, hence completing the Local Pattern Discovery phase.

### 9.3 Pattern Subset Discovery Phase

Having outlined the details of local pattern discovery in a multilabel context, we now proceed to the second phase of the LeGo-framework: Pattern Subset Discovery. A common approach for *feature subset selection* for regular classification problems is to measure some type of correlation between a feature variable and the class variable. A subset of the features  $\mathcal{S}$  from the whole set  $\mathcal{P}$  is then determined either by selecting a predetermined number of best features or by selecting all features whose value exceeds a predetermined threshold. Unfortunately, this approach is not directly applicable to multilabeled data without adaptation. Several solutions are possible, we experimented with the following approaches.

A simple way is to convert the multilabel problem into a *multiclass* (MC) classification problem, where each original instance is converted into several new instances, one for each label  $\lambda_i$  it belongs to, using  $\lambda_i$  as the class value (cf. Figure 9.2(b)). However, this transformation does not take the underlying decomposition into account.

An alternative approach is to measure the correlations on the decomposed subproblems produced by the *binary relevance decomposition* (cf. Section 3.1). The  $n$  different correlation values for each feature are then aggregated. In our experiments, we aggregated with the max operator, i.e., the overall relevancy of a feature was determined by its maximum relevance in one of the training sets of the binary relevance classifiers. The main drawback of this approach is that it treats all labels independently and ignores that a features might only be relevant for a combination of class labels, but not for the individual labels.

The last approach employs the *label powerset* (LP) transformation (cf. Section 3.2) in order to also measure the correlation of a feature to the simultaneous absence or occurrence of label sets. Hence, with the dataset transformed into a multiclass problem, common feature selection techniques can be applied. The different decomposition approaches for the feature selection are depicted in Figure 9.2.

After the transformations, we can use common attribute correlation measures for evaluating the importance of an attribute in each of the three approaches. In particular, we employed the information gain and the chi-squared statistics value of an attribute with respect to the class variable resulting from the particular decomposition method applied. Hence, the statistic is computed between each of the  $a$  attribute columns  $(x_{i,j})_i$  and the last column in Figures 9.2(b), 9.2(c) or 9.2(d). Then we let each of the six feature selection methods select the best patterns from  $\mathcal{P}$  to form the subset  $\mathcal{S}$ . The size  $|\mathcal{S}|$  of the subset is fixed in our experiments (cf. Section 9.4).

Attributes	Labels
$x_{1,1}, \dots, x_{1,a}$	$y_{1,1}, \dots, y_{1,n}$
$x_{2,1}, \dots, x_{2,a}$	$y_{2,1}, \dots, y_{2,n}$
$\vdots \quad \ddots \quad \vdots$	$\vdots \quad \ddots \quad \vdots$
$x_{m,1}, \dots, x_{m,a}$	$y_{m,1}, \dots, y_{m,n}$

(a) input training set  $h_{\mathcal{O}}$

Attributes	Labels
$p_1(x_{1,1}, \dots, x_{1,a}), \dots, p_{ S }(x_{1,1}, \dots, x_{1,a})$	$y_{1,1}, \dots, y_{1,n}$
$p_1(x_{2,1}, \dots, x_{2,a}), \dots, p_{ S }(x_{2,1}, \dots, x_{2,a})$	$y_{2,1}, \dots, y_{2,n}$
$\vdots \quad \ddots \quad \vdots$	$\vdots \quad \ddots \quad \vdots$
$p_1(x_{m,1}, \dots, x_{m,a}), \dots, p_{ S }(x_{m,1}, \dots, x_{m,a})$	$y_{m,1}, \dots, y_{m,n}$

(b) transformation into pattern space  $h_{\mathcal{S}}$

**Figure 9.4:** A multilabel classification problem (a) and its representation in pattern space (b), given the set of patterns  $p_1, \dots, p_{|S|}$ .

The approach adapted from multiclass classification of measuring the correlation between each feature and the class variable has known weaknesses such as being susceptible to redundancies within the features. Hence, in order to evaluate the feature selection methods, we will compare them with the baseline method that simply draws  $\mathcal{S}$  as a random sample from  $\mathcal{P}$ .

## 9.4 Global Modeling Phase

For the learning of the global multilabel classification models in the Global Modeling phase, we experiment with several standard approaches including binary relevance and label powerset transformations as well as with the recent effective state-of-the-art learner classifier chains (CC) (Read et al. 2009) and our calibrated pairwise approach (CLR) (cf. Section 3.4.5). The chosen algorithms cover a wide range of different approaches and techniques used for learning multilabel problems and are all included Mulan (Tsoumakas et al. 2011b).

For each classifier configuration, we learn three classifiers based on different feature sets. The first classifier uses only the  $a$  features that make up the original dataset, and is denoted  $h_{\mathcal{O}}$  (Figure 9.4(a)). The second classifier uses only features constructed from our pattern set  $\mathcal{S}$ , and is denoted  $h_{\mathcal{S}}$ . Each of these patterns by definition maps each record in the original dataset to either zero or one. Hence they can be trivially transformed into binary features, that together make up the feature space for classifier  $h_{\mathcal{S}}$  (Figure 9.4(b)). The third classifier employs both original and constructed features, in the spirit of LeGo, and is hence denoted  $h_{\mathcal{L}}$ . Its feature space consists of the  $a$  original features, and  $|S|$  features constructed from the pattern set  $\mathcal{S}$  for a grand total of  $a + |S|$  features.



**Table 9.1:** Datasets used in the experiments, shown with the number of examples ( $m$ ), attributes ( $a$ ), and labels ( $n$ ), as well as the average number of labels per example ( $d$ ).

Dataset	Domain	$m$	$a$	$n$	$d$
<i>emotions</i>	music	593	72	6	1.87
<i>scene</i>	vision	2407	294	6	1.07
<i>yeast</i>	biology	2417	103	14	4.24

## 9.5 Experimental Setup

To experimentally validate the outlined LeGo method, we will compare the performance of the three classifiers based on different feature sets  $h_{\mathcal{O}}$ ,  $h_{\mathcal{S}}$ , and  $h_{\mathcal{L}}$ . We will also investigate the relative performance of the different feature selection methods, and the relative performance of classification approaches.

### 9.5.1 Datasets and Learners

For the experiments we selected three popular multilabel datasets from different domains (cf. Section 2.9.1). The basic statistics on these datasets can be found in Table 9.1.

We combine the multilabel decomposition methods mentioned in Section 9.4 with several base learners: J48 with default settings (Witten and Frank 2005), standard LibSVM (Chang and Lin 2001, see also Section 4.1.6), and LibSVM with a grid search on the parameters. In this last approach, multiple values for the SVM kernel parameters are tried, and the one with the best 3-fold cross-validation accuracy is selected for learning on the training set (as suggested by Chang and Lin 2001). Both SVM methods are run once with the Gaussian Radial Basis Function as kernel, and once with a linear kernel using the efficient LibLINEAR implementation (Fan et al. 2008). We will refer to LibSVM with the parameter grid search as MetaLibSVM, and denote the used kernel by a superscript *rbf* or *lin*.

### 9.5.2 Obtaining Raw Results

All statistics on the classification processes are estimated via a 10-fold cross-validation. To enable a fair comparison of the LeGo classifier with the other classifiers, we let the entire learning process consider only the training set for each fold. This means that we have to run the Local Pattern Discovery and Pattern Subset Discovery phase separately for each fold. Parameters are adjusted with an internal cross-validation on the training data (see above).

For every fold on every dataset, we determine the best 10,000 patterns. The beam search was configured with a beam width of  $w = 10$  and a maximum search level of 2, i.e. every pattern corresponds to at most two restrictions on one attribute each. We specifically select a search of modest depth, in order to prevent producing an abun-



**Table 9.2:** Average ranks of different feature selection methods, with critical difference.

Selection method	BR		LP		MC		Random	CD
Evaluation method	$\chi^2$	gain	$\chi^2$	gain	$\chi^2$	gain		
Rank	4.445	3.932	3.507	4.263	3.707	4.490	3.657	0.520

dance of highly similar patterns. We further bound the search by setting the minimal coverage of a pattern at 10% of the dataset.

Notice that the choice of search lattice traversal does not influence the way the exceptionality measure determines the quality of a pattern. While setting the search level to 2 does influence the complexity of the pattern in attribute space, this setting is independent from the complexity of the models we fit in label space. Each candidate pattern will be evaluated by fitting a Bayesian network to all its labels, regardless of search parameters such as the search level.

For each dataset for each fold, we train classifiers from the three training sets  $h_O, h_S$ , and  $h_L$  for each combination of a decomposition approach and base learner. In the Pattern Subset Discovery phase, we set  $|\mathcal{S}| := a$ , i.e. we select exactly as many pattern-based features for  $\mathcal{S}$  and  $h_L$  as there are original features in  $h_O$ .

As evaluation measures we chose (label and example-based) micro-averaged precision, recall, subset accuracy, ranking loss and average precision. We find these five measures a well balanced selection from the vast set of multilabel measures, evaluating different aspects of multilabel predictions such as good ranking performance and correct bipartition (cf. Section 2.7).

All values for all settings are averaged over the folds of the cross-validation. Thus we obtain a grand total of 300 test cases (5 evaluation measures  $\times$  5 base learners  $\times$  4 decomposition approaches  $\times$  3 datasets).

To draw conclusions from the long list of raw results we obtained, we use the methodology of combined Friedman and Nemenyi test (cf. Section 2.10.3) with a significance level of  $\alpha = 5\%$ . Depending on the combinations we compare, we perform the tests on different numbers of samples  $N$  and approaches  $k$ . The Friedman test was passed in all evaluated cases.

## 9.6 Evaluation

The following subsections are dedicated to different aspects such as the evaluation of the different pattern subset discovery approaches, the employment of the different feature sets, the impact of the decomposition approaches, and efficiency.

---

### 9.6.1 Feature Selection Methods

Before comparing the three classifiers, we take a look at the relative performance of the different feature selection methods. When comparing the performance of the classifier  $h_{\mathcal{L}}$  with different feature selection methods over all  $N = 300$  test cases, we find the average ranks in Table 9.2. We compared the binary relevance, label powerset and multiclass approach, each with evaluation measures chi-squared and information gain, and the random baseline approach.

The results show that no classifier employing a sophisticated feature selection method significantly<sup>64</sup> outperforms the classifier with random feature selection, which in turn does significantly outperform several classifiers employing sophisticated feature selection. For the binary relevance and multiclass approaches this is reasonable, since the patterns are explicitly designed to consider interdependencies between labels, while the BR and MC approaches select features based on their correlation with single labels only and hence totally ignore interdependencies. The label powerset approach should do better in this respect. In fact, the best average rank featured in Table 9.2 belongs to LP with the chi-squared evaluation measure. However, since its improvement over the naive method is not significant, we did not further explore its performance.

Another reason for the bad performance of the the feature selection methods is that they evaluate each feature individually. One extreme use case will show the problem: if we replicate each feature  $n$  times and we select the  $n$  best features according to the presented methods, we will get  $n$  times the same (best) feature. In the Local Pattern Mining phase, we produce a high number of additional features, hence we hardly can avoid to obtain groups of similar additional features where this problem may appear. The random feature selection does not suffer from this problem. As a result of these experiments, we decided not to use any sophisticated feature selection in the remaining experiments, and focus on the results for random feature selection.

### 9.6.2 Evaluation of the LeGo Approach

The first row in Table 9.3 compares the three different classifiers  $h_{\mathcal{O}}$ ,  $h_{\mathcal{S}}$  and  $h_{\mathcal{L}}$  trained on the three different feature representations in terms of average ranks over the grand total of 300 test cases: four decomposition methods BR, CC, CLR, and LP, combined with five base learners, tested on the three datasets from Table 9.1, evaluated with the five measures specified in Section 9.5.2. We see that both  $h_{\mathcal{O}}$  and  $h_{\mathcal{L}}$  perform significantly better than  $h_{\mathcal{S}}$ , i.e. the pattern-only classifier cannot compete with the original features or the combined classifier.

The difference in performance between  $h_{\mathcal{O}}$  and  $h_{\mathcal{L}}$  is not significant. Although the average rank for the LeGo-based classifier is somewhat higher, we cannot claim that adding local patterns leads to a significant improvement. However, when splitting out the results for the different base learners (the second block in Table 9.3), we notice a

---

<sup>64</sup> The Friedman test with  $N = 300, k = 7$  is comfortably passed for  $\alpha = 5\%$ . Nemenyi's critical distance becomes  $CD = 0.520$  requiring  $\alpha = 5\%$  (cf. Section 2.10.3).

**Table 9.3:** Comparison of different feature sets. Average ranks of the three classifiers  $h_O$ ,  $h_S$ ,  $h_L$ , with critical difference, over all 300 test cases, over all 240 test cases barring J48, over all 60 test cases with a particular base learner, and over all 75 test cases with a particular decomposition method. Bold numbers indicate the top rank in the row, > or < indicates a significant difference to the direct neighbor classifier.

	$h_O$	$h_L$	$h_S$	CD
Overall	1.863 = <b>1.797</b> > 2.340			0.191
Without J48	1.971 < <b>1.733</b> > 2.296			0.214
MetaLibSVM <sup>rbf</sup>	<b>1.483</b> = 1.683 > 2.833			0.428
MetaLibSVM <sup>lin</sup>	1.900 = <b>1.800</b> > 2.300			"
LibSVM <sup>rbf</sup>	2.633 < <b>1.683</b> = <b>1.683</b>			"
LibSVM <sup>lin</sup>	1.867 = <b>1.767</b> > 2.367			"
J48	<b>1.433</b> > 2.050 > 2.517			"
CLR	1.813 = <b>1.760</b> > 2.427			0.383
LP	<b>1.773</b> = 1.827 > 2.400			"
CC	1.947 = <b>1.720</b> > 2.333			"
BR	1.920 = <b>1.880</b> = 2.200			—

striking difference in average ranks between J48 and the rest. When we restrict ourselves to the results obtained with J48, we find that the classifier built from original features significantly outperforms the LeGo classifier.

One reason for the performance gap between J48 and the SVM approach lies in the way these approaches construct their decision boundary. The SVM approaches draw one hyperplane through the attribute space, whereas J48 constructs a decision tree, which corresponds to a decision boundary consisting of axis-parallel fragments. Now the patterns the EMM algorithm finds in the Local Pattern Discovery phase are constructed by several conditions on single attributes. Hence the domain of each pattern has a shape similar to a J48 decision boundary, unlike a (non-degenerate) SVM decision boundary. Hence the expected performance gain when adding such local patterns to the attribute space is much higher for the SVM approaches than for the J48 approach.

However, using only the original features seems to be also enough for the highly optimized non-linear SVM, though the difference to the combined features is small and not statistically significant. The remaining base learner benefit from the additional local patterns. Notably, when using LibSVM<sup>rbf</sup>, it is even possible to rely only on the pattern-based features in order to outperform the classifiers trained on the original features.

Because particularly the J48 approach results in such deviating ranks, we investigate the relative performance of the base learners. We compare their performance on the three classifiers  $h_O$ ,  $h_S$ , and  $h_L$ , with decomposition methods BR, CC, CLR, and LP, on the datasets from Table 9.1, evaluated with the measures specified in Section 9.5.2. The average ranks of the base learners over these 180 test cases can be found in Table 9.4. Again, the Friedman test is easily passed, and the Nemenyi test shows that J48 performs

**Table 9.4:** Comparison of the different base learners in terms of the average ranks on 180 test cases, respectively, with critical difference.

Approach	Rank
MetaLibSVM <sup>rbf</sup>	1.489
MetaLibSVM <sup>lin</sup>	2.972
LibSVM <sup>lin</sup>	3.228
LibSVM <sup>rbf</sup>	3.417
J48	3.894
CD	0.455

significantly worse than all SVM methods, and that MetaLibSVM<sup>rbf</sup> clearly dominates the performance of the SVMs. This last point is not surprising, since the three datasets are known to be difficult and hence not linearly separable (cf. Section 4.6.8), which means that an advantage of the RBF-kernel over the linear kernel can be expected. Moreover, the non expensively optimized LibSVM<sup>rbf</sup> can be considered to be subsumed by the meta variant since the grid search also tries out the default settings.

Having just established that J48 is the worst-performing base learner and, additionally, that the similar decision patterns particularly damage the performance of the LeGo classifier, we repeat our overall comparison considering only the SVM variants. Moreover, SVMs are conceptually different from decision tree learners, which additionally justifies the separate comparison. The average ranks of the three classifiers  $h_{\mathcal{O}}$ ,  $h_{\mathcal{S}}$ , and  $h_{\mathcal{L}}$  on the remaining 240 test cases can be found in the second row of Table 9.3. This time, the Nemenyi test yields that on the SVM methods the LeGo classifier is generally significantly better than the classifier built from original features, even though for MetaLibSVM<sup>rbf</sup> this is not the case.

The last block in Table 9.3 allows a further differentiation with respect to the employed transformation technique. With the exception of the label powerset approach, which already maximally respects label dependencies, all approaches benefit from the combination with the constructed LeGo features, though the differences are not statistically significant. Of peculiar interest is the benefit for the binary relevance approach, which in its original form considers each label independently. Though the Friedman test is not passed, the trend is confirmed by the results of CC, which additionally include features from the preceding base classifiers' predictions.

If we consider CC not as a different decomposition approach but as a feature enriching approach, which adds the feature set  $\mathcal{C}$ , the result comparing the performance of the different feature sets arrives at  $\mathcal{O} \cup \mathcal{S} \cup \mathcal{C}$  (rank 2.84) followed by  $\mathcal{O} \cup \mathcal{S}$  (3.34),  $\mathcal{O} \cup \mathcal{C}$  (3.53),  $\mathcal{O}$  (3.6),  $\mathcal{S}$  (3.89) and  $\mathcal{S} \cup \mathcal{C}$  (3.97) (significant difference only between first and both last combinations). Hence, adding  $\mathcal{C}$  has a similar effect on performance than adding  $\mathcal{S}$ , and BR particularly benefits if both are added, which demonstrates that the patterns based on local exceptionalities provide additional information on the label dependencies which is not covered by  $\mathcal{C}$ .

**Table 9.5:** Comparison of the decomposition approaches. The first block compares the approaches for all base learner combinations, the second one restricts on the usage of MetaLibSVM<sup>rbf</sup>. The first row in each block indicates the average ranks with respect to all evaluation metrics, whereas the following rows distinguish between the individual measures.

Measure	CLR	LP	CC	BR	CD
all & all BC	<b>1.909</b>	> 2.462	= 2.700	= 2.929	0.313
ACC	3.400	< <b>1.489</b>	= 1.722	> 3.389	0.700
PREC	1.989	> 3.467	= 3.111	< <b>1.433</b>	"
REC	2.156	= <b>1.956</b>	= 2.422	> 3.467	"
AvGP	<b>1.000</b>	> 2.778	= 3.111	= 3.111	"
RANKLOSS	<b>1.000</b>	> 2.622	= 3.133	= 3.244	"
all & MetaLibSVM <sup>rbf</sup>	2.111	= <b>1.911</b>	> 2.911	= 3.067	0.700
ACC	3.667	< <b>1.000</b>	= 2.000	= 3.333	1.563
PREC	2.111	= 3.444	= 3.444	< <b>1.000</b>	"
REC	2.778	< <b>1.000</b>	= 2.333	= 3.889	"
AvGP	<b>1.000</b>	= 2.111	= 3.444	= 3.444	"
RANKLOSS	<b>1.000</b>	= 2.000	= 3.333	= 3.667	"

### 9.6.3 Evaluation of the Decompositive Approaches

In the previous results shown in Table 9.3 there has been no differentiation between the different evaluation measures. Indeed, the decision of the feature base does not seem to have a differing impact on the metrics (for the SVM learners, not shown in the table). The only exception appears to be micro-averaged precision, for which  $h_O$  yields a small advantage over  $h_C$ . But as Table 9.5 demonstrates, the situation changes with respect to the decompositive approach used. As we can see in the upper block, there are clear tendencies regarding the preference for a particular metric.

LP e.g. has a clear advantage in terms of subset accuracy, which only CC is able to catch up. This is natural, since both approaches are dedicated to the prediction of a correct labelset. In fact, LP only predicts labelsets previously seen in the training data. CC behaves similarly, as is shown in the following: If we consider only the additional features from the previous predictions (i.e. feature set  $C$ ), then we find that CC behaves like a sequence tagger. Hence, for a particular sequence of labels  $y_1, \dots, y_{i-1}$  the  $i$ -th classifier in the chain will tend to predict  $y_i = 1$  (or  $y_i = 0$  respectively) only if  $y_1, \dots, y_i$  existed in the training data (see also Section 3.5.7 for the detailed functioning of CC). The advantage of LP and CC is confirmed in the bottom block, which restricts the comparison to the usage of the most accurate base learner MetaLibSVM<sup>rbf</sup>.

Precision is dominated by BR, followed by CLR. This result is obtained by being very cautious at prediction, as the values for recall show. Especially the highly optimized SVM is apparently fitted towards predicting a label only if it is very confident that the esti-

mation is correct. Whether this is because of the high imbalance of the binary subproblems, e.g. compared to pairwise decomposition, is not clear, but it again demonstrates the problematic issue of optimizing the base learners in BR, which was already discussed in Section 3.5.7 and 3.5.4. CLR shows to be more robust, though the bias towards underestimating the size of the labelsets (cf. Section 4.6.9) is visible. Especially in this case the bias may originate from the conservative BR classifiers, which are included in the calibrated ensemble, since the difference between precision and recall is clearly higher for  $\text{MetaLibSVM}^{\text{rbf}}$ .

A contrary behavior to BR is shown by LP, which dominates recall, especially for  $\text{MetaLibSVM}^{\text{rbf}}$ , but completely neglects precision. This indicates a preference for predicting the more rare large labelsets. The best balance between precision and recall is shown by CLR. Even for  $\text{MetaLibSVM}^{\text{rbf}}$ , for which the underestimation leads to low recall, but for which the competing classifier chains obtain the worst precision values together with LP.

The good balancing properties of CLR are confirmed by the results on the ranking losses, which are clearly dominated by CLR's good ability to produce a high density of relevant labels at the top of the the rankings. The high recall of LP is corresponded with good ranking losses, but the low ranks of BR demonstrate that its high precision is not due to a good ranking ability. This behavior was already observed on Reuters *rcv1* in Section 4.6.2 and in on *EUR-Lex* Section 6.4.2, where BR often correctly pushed a relevant class at the top ( $\text{ONEERR}$ ), but obtained poor ranking losses. Similarly, CC's base classifiers are trained independently without a common basis for confidence scores and hence achieve a low ranking quality. The technique of Fan and Lin (2007) described in Section 3.5.4 may alleviate this disadvantage at the expense of additional computational costs, a measure which is obviously not necessary for CLR (cf. Section 3.5.6).

If we give the same weight to the five selected measures, we observe that CLR significantly outperforms the second placed LP if all base learners are considered and slightly loses against LP if  $\text{MetaLibSVM}^{\text{rbf}}$  is used (top row in both blocks in Table 9.3).

#### 9.6.4 Efficiency

Apart from the unsatisfactory performance of the J48 approach compared to SVM approaches, Table 9.4 also indicates that compared to the standard LibSVM approach the extra computation time invested in the  $\text{MetaLibSVM}$  parameter grid search is rewarded with a significant increase in classifier performance. For both the linear and the RBF kernel we see that the  $\text{MetaLibSVM}$  approach outperforms the LibSVM approach, although this difference is only significant for the RBF kernel. A more exhaustive parameter-optimizing search will probably be beneficial, since the grid search considers arbitrary parameter values. Whether the performance increase is worth the invested time is a question of preference. In the case where time and computing power are not limited resources, the increased performance is clearly worthwhile, though an overfitting is visible which especially affects BR.

From a practical point of view it is also interesting to analyze the efficiency of using the original features in comparison to using the constructed features. We expected an improvement in complexity just from the circumstance that the pattern features are binary in contrast to the more complex nature of the original features in the used datasets. The gain may not only originate from the fact that binary features are far easier to process than discrete or even continuous data for a large range of learners such as decision trees and rule learner, but also from the possible resulting feature value sparseness which also boosts algorithms like SVMs.

For the comparison between training of  $h_{\mathcal{O}}$  and  $h_{\mathcal{S}}$  we focus on the binary relevance decomposition setting in order to allow a balanced comparison over the three datasets, since the complexity of BR scales linearly with respect to the number of classes. For J48 as base learner, we observed a reduction of training costs from 28% (*emotions*) over 31% (*scene*) to 47% for *yeast*. For  $\text{LibSVM}^{\text{rbf}}$ , the difference was more pronounced, with a reduction of 60%, 52% and 50%, respectively. We obtained a similar picture for  $\text{LibSVM}^{\text{lin}}$ , with a reduction of even 82% (*emotions*) and 65% (*scene*), except for *yeast*, where training  $h_{\mathcal{S}}$  surprisingly takes almost 7 times longer than training  $h_{\mathcal{O}}$ . This case is very likely an exception since comparing  $\text{MetaLibSVM}^{\text{lin}}$  and hence using different parameters shows again a clear reduction. The numbers for  $\text{MetaLibSVM}^{\text{lin}}$  and  $\text{MetaLibSVM}^{\text{rbf}}$  are omitted since they show a similar picture but are more difficult to compare directly since they always also include testing time.

Note that training  $h_{\mathcal{L}}$  of course takes more computation time since we employ both feature sets, but the overhead of using the constructed features from the patterns is clearly relatively small. Also, notice that the overhead of producing the features needs to be invested only once for training the classifier, possibly off-line, and that the resulting trained classifier can then be used again and again for classifying data; if one wants to classify more than once, the added costs diminishes.

## 9.7 Discussion and Related Work

The applied Local Pattern Discovery algorithm was created to find patterns that are interesting by themselves.<sup>65</sup> The output of the algorithm is therefore not specifically tailored to be useful in a classification setting; this is not a guiding principle in the Exceptional Model Mining process. To the best of our knowledge, this work is a first shot at testing the utility for classification of the result of such a stand-alone multilabel pattern discovery process. Some recent sophisticated classifiers, for instance the multilabel lazy associative classifiers (Veloso et al. 2007), are also based on local patterns. However, these patterns serve only the classifier, hence the different phases, as present in the LeGo framework, are not as separated as they are in our work. Similarly, Cheng and Hüllermeier (2009) incorporate additional features that encode the label distribution in the direct neighborhood by in effect stacking the output of a  $k$ -Nearest Neighbor classi-

<sup>65</sup> We omit the related work on subgroup discovery since this is not the main concern in the frame of this thesis. Nevertheless, we refer the interested reader to Duivesteijn et al. (2012,?).



---

fier (cf. [Witten and Frank 2005](#), Sec. 3.8). However, this has to be done at (training and testing) runtime and cannot be done separately and beforehand.

Other known stacking approaches include the outcome of global classifiers. [Godbole and Sarawagi \(2004\)](#) e.g. use the outputs of a BR-SVM classifier as additional input features for second level SVMs. Similarly, [Tsoumakas et al. \(2009a\)](#) replace all original features by the predicted scores of a BR. The scores are additionally filtered according to their correlation to each other. Classifier chains ([Read et al. 2009](#)) rely on stacking the outcomes of the predetermined sequence of previous binary relevance classifiers, which permits to model conditional dependencies, but it does not rely on locality. [Zhang and Zhang \(2010\)](#) also try to model label dependencies and start from the premise of eliminating the conditional dependency between the input features  $\mathbf{x}$  and the individual labels by computing the errors  $e_i = y_i - h_i(\mathbf{x})$ . The isolated dependencies between labels are then approximated by the result of building a Bayesian network on these errors. A new BR classifier is then learned for each class with the set of alleged parents as additional features. The very recent LIFT algorithm selects particularly representative centroids in the positive and negative examples of a class by  $k$ -means clustering (cf [Witten and Frank 2005](#), Sec. 4.7) and then replaces the original features of an instance by the distances to these representatives ([Zhang 2011](#)). One may also interpret this approach as a different, pragmatic way of computing new suitable principal components and hence dimensionality reduction, but which apparently works quite well.

## 9.8 Summary

We have proposed to enhance multilabel classification methods with local patterns in a LeGo setting. These patterns are found through an instance of Exceptional Model Mining, a generalization of Subgroup Discovery striving to find subsets of the data with aberrant conditional dependence relations between target features. Hence each pattern delivered by this method represents a local anomaly in conditional dependence relations between the labels. Each pattern corresponds to a binary feature which we add to the dataset, in order to improve classifier performance.

Experiments on three datasets show that for multilabel SVM classifiers the performance of the LeGo approach is significantly better than the traditional classification performance: the statistical analysis shows that investing extra time in running the EMM algorithm pays off when the resulting patterns are used as constructed features. The J48 classifier does not benefit from the addition of local patterns, but it generally performs significantly worse than the SVM-based classifiers. This can be attributed to the similarity of the local decision boundaries produced by the subgroup discovery algorithm to those produced by the decision tree learner. Hence, the expected performance gain when adding local patterns is lower for J48 than for approaches that learn different types of decision boundaries, such as SVM approaches.

From the point of view of pairwise multilabel learning, the presented technique for exploring label dependencies demonstrates that the restriction on detecting label relations of second order can potentially be surmounted, e.g., simply by a feature enrichment pro-



---

cess such as the one presented. This is of course not limited to pairwise decomposition but also applies to any otherwise correlation blind approach. The local exploration ensures that the interesting conditional dependencies are taken into account.

The Friedman-Nemenyi analysis also shows that the constructed features cannot *replace* the original features without significant loss in classification performance. We find this reasonable, since these features are constructed from patterns found by a search process that is not at all concerned with the potential of the patterns for classification, but is focused on exceptionality. In fact, the pattern set may be highly redundant. Hence it is likely that the less exceptional part of the data, which by definition is the majority of the dataset, is underrepresented by the constructed features.

As mentioned in Section 9.7, to the best of our knowledge, this is a first shot at discovering multilabel patterns and testing their utility for classification in a LeGo setting. Therefore this work can be extended in various ways. It might be interesting to develop more efficient techniques without losing performance. It seems fruitful to investigate alternative search strategies which rely on different quality measures or are focused on properly balancing its levels of exploration and exploitation.

Such a modified search strategy would possibly produce a more diverse set of features and avoid some of the observed redundancy issues in the current top-10,000 patterns. Focusing on feature selection, these issues might also be addressed in future work by adapting the feature evaluation in the Pattern Subset Selection phase such that diversity within the subset is considered. This could then possibly lead to the targeted full replacement of the original features, enabling more efficient learning and classification.

Regarding the clear preferences of the different decomposition approaches towards a particular evaluation metric in our experiments in Section 9.6.3, there is a strong motivation to further investigate how to adapt the aggregation process of the pairwise ensemble in order to maximize a particular arbitrary measure. Whereas the maximization of recall or precision seems straightforward, optimizing the combined  $F$ -measure or the subset accuracy is less obvious. Intuitively, improving ranking loss and average precision will also improve the maximally obtainable F1 score,<sup>66</sup> and simple voting (cf. Section 3.4.3) evidently achieves exactly this. But the maximization of subset accuracy apparently requires alternative aggregation strategies such as ranking through iterated choice (Hüllermeier and Fürnkranz 2007) or the technique of Park and Fürnkranz (2008) for the correction of rankings considering previously observed label constellations (see Section 3.5.5 for further explanations).

---

<sup>66</sup> Unfortunately, the employed framework for the experiments (Tsoumakas et al. 2011b) does not measure the idealistic  $F1_{|p|}$  (cf. Section 2.7.4). We plan to consider this in future evaluations.



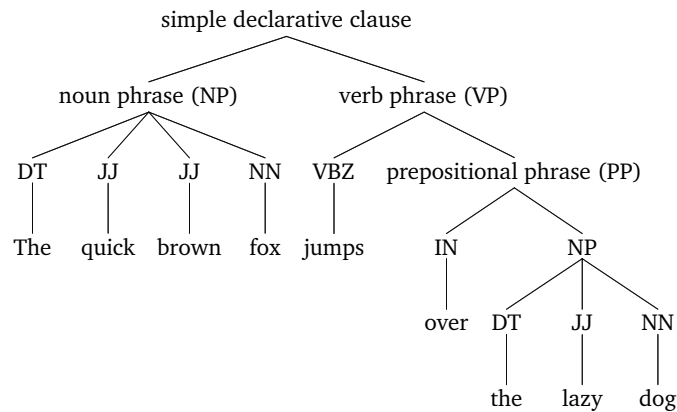
## 10 Information Extraction and Syntactic Parsing

In recent years, more and more approaches have appeared that translate the *information extraction* task into a classical classification problem, which is nowadays considered the most popular approach. The most common approach is to transform each text position, i.e. usually each text token in the document, into a classification example. This is often called boundary classification or sequential tagging/labeling. The class information of the instance depends on whether the underlying text token is a part or boundary of the target annotation or not. Figure 10.1 shows an tagged sentence, whose syntax tree is depicted in Figure 10.2. The token *The* is marked as the beginning of a noun phrase (*[NP]* that ends at *fox* with *NP]*. The standard approach is to train one separate classifier for each annotation type, i.e. one for noun phrases, one for verb phrases etc. This subproblems can be solved using a multiclass classification algorithm that is trained to predict for each token exactly one class. We can often observe an overlapping of the annotations of the different types in real world applications, such as in chunking, syntactic parsing or ontology based information extraction (OBIE) (cf. Li and Bontcheva 2007). The token *The* e.g. is at the same time a determiner and the beginning of a noun phrase, which is indeed linguistically a reasonable coincidence. The traditional approach ignores this co-occurrence and is therefore not able to exploit the additional information, namely that determiners are often also noun phrases beginnings.

The approach presented in this chapter therefore reformulates the many individual multiclass problems into only one multilabel classification problem. This means that a token is now allowed to have simultaneously several classes assigned. The reason for this representation is twofold: on the one hand we obtain a more natural, compact and con-

token	The	quick	brown	fox	jumps	over	the	lazy	dog
token	the=1	quick=1	brown=1	fox=1	jumps=1	over=1	the=1	lazy=1	dog=1
features	+1.quick=1	+1.brown=1	+1.fox=1	+1.jumps=1	+1.over=1	+1.the=1	+1.lazy=1	+1.dog=1	
		-1.the=1	-1.quick=1	-1.brown=1	-1.fox=1	-1.jumps=1	-1.over=1	-1.the=1	-1.lazy=1
POS	[DT, DT]	[JJ, JJ]	[JJ, JJ]	[NN, NN]	[VBZ, VBZ]	[IN, IN]	[DT, DT]	[JJ, JJ]	[NN, NN]
syntactic	[NP			NP]	[VP	[PP	[NP		NP], PP], VP]

**Figure 10.1:** Transformation of a text sentence into a classification problem. Each column shows a token and exemplarily the generated features with a context of one word and the class information of the corresponding generated classification instance. The first row of the class information 'POS' shows the part-of-speech annotations, the second the syntactic annotations given to the token. Abbreviations according to Marcus et al. (1993). A '[' denotes the *BEGIN* and ']' the *END* of the corresponding annotation type.



**Figure 10.2:** Example of a syntax tree. The result was generated with the Stanford Parser (cf. Section 10.3).

sistent representation of the extraction problem. On the other hand, the main purpose of this formulation is to allow an underlying multilabel algorithm to exploit relations in the labels such as co-occurrence, exclusions and implications and hence improve the prediction quality. This work evaluates several multilabel learning algorithms and compares them on a dense annotation dataset.

## 10.1 Information Extraction

The transformation of an information extraction task into a classification problem was already sketched in the introduction. The next two subsections give a more detailed description of the two main processing steps (mainly based on [Loza Mencía 2009](#)).

### 10.1.1 Boundary Classification

In this work we employ the simple but effective *Begin/End* approach. The start and the end of each annotation, i.e. only the boundaries, are marked with the tag *BEGIN* or *END*, the rest is marked as negative examples with *NEG*. The bottom two rows in Figure 10.1 shows for each type DT (determiner), JJ (adjective), NN (noun) etc. (cf. [Marcus et al. 1993](#) for abbreviations) the beginning and ending of an annotation.<sup>67</sup>

In the standard approach, a problem appears when an annotation only includes one token, such as for DT. This would make it necessary to tag a token as *BEGIN* and *END* simultaneously. As this would require a multilabel capable underlying classifier, the common approach is to include an additional class *UNIQUE* which represents both classes at the same time (see also Section 10.2).

<sup>67</sup> The negative class is omitted since this is the multilabel representation.

---

We chose a pragmatic way for solving inconsistencies during the reconstruction of the annotations on the test set: we search for the first appearing of an opening tag and continue the extraction until the first matching closing tag is found, the remaining tags until then are simply ignored.

### 10.1.2 Feature Generation

The boundary classification step generates the class information for each training instance, but up to now these instances are empty. The simplest features which can be added are the occurrences of the different tokens themselves. Since the focus of this work lies on the comparison of the classification algorithms, we use only these simple features and refrain from using sophisticated linguistic features. For the same reason we ignore the classification history.

Windowing ensures that the context of the current text position is taken into account. A predetermined number of preceding and following tokens coded together with their position are used (see the token features row in Figure 10.1 for a window size of one).

## 10.2 Multilabel Classification for Information Extraction

As already outlined, one of the advantages of multilabel classification is the more natural representation since we do not have to work with tricks. Note e.g. that in the *BEGIN/END/UNIQUE* scheme the learning algorithm is forced to learn to distinguish between *UNIQUE* and *BEGIN* or *END*, respectively, though *UNIQUE* is actually a subset of these two classes. This could lead to a deterioration of the detection performance for the start and end boundaries. This makes this tagging scheme especially interesting for the usage of multilabel classification. Furthermore, the traditional methods do not permit to exploit relations between several annotation types since each type is by design necessarily learned separately. We present therefore in the following the standard approach together with the multilabel alternatives.

### Traditional Multiclass Approach

A multiclass classifier is trained for each annotation type, the *BEGIN/END/UNIQUE/NEG* scheme is used. I.e. for each annotation type  $t \in \mathcal{A}$  a classifier is trained with instances mapped to exactly one class  $\lambda$  in  $\mathcal{L}^{(t)} = \{\textit{BEGIN}, \textit{END}, \textit{UNIQUE}, \textit{NEG}\}$ . The multiclass problem is solved via one-against-all decomposition in our case (MC-BR).

### Binary Relevance

The extraction task is transformed into one multilabel problem where each token is assigned to a subset  $P$  of  $\mathcal{L} = \mathcal{A} \times \{\textit{BEGIN}, \textit{END}\}$ , with  $\mathcal{A}$  as the set of annotation types. Note that it is not necessary to include neither *NEG* nor *UNIQUE* since we are able to predict the empty set as well as *BEGIN* and *END* simultaneously.

---

The binary relevance approach (cf. Section 3.1) learns a set of classifiers  $h_{t,B}$ ,  $h_{t,E}$ ,  $t \in \mathcal{A}$  that each predicts whether an annotation  $t$  begins (label  $t, B$ ) or ends (label  $t, E$ ).

### Pairwise Decomposition

In the binary relevance setting, the algorithm is not expected to improve from label co-occurrences since each base classifier is trained separately. However, the pairwise approach is at least potentially able to detect non co-occurrences, since we train for each pair of classes a base classifier with instances where the one class is positive and the other negative, i.e. the base classifier is trained with cases where both classes are mutually exclusive (cf. Section 8.3.1). Therefore this approach may be able detect that a co-prediction of two classes is wrong for a determined instance, in contrast to BR, where a base classifier cannot state anything else than relevant or non-relevant.

We employ calibration (cf. Section 3.4.5) in order to predict bipartitions and QVoting (cf. Section 5.1) in order to boost the extraction process (QCLR).

### Label Powerset

In the label powerset approach (LP), a meta multiclass problem is constructed where each appearing label combination  $P_i$  is interpreted as one separate class. The meta problem is then solved with a normal multiclass algorithm or with the previously presented decomposition methods (cf. Section 3.2).

The multilabel problem is re-transformed into a multiclass problem, i.e. the possible classes of a token are in  $\mathcal{L} = 2^{A \times \{BEGIN, END\}}$ . Note that for only one annotation type this corresponds to the traditional multiclass approach, since we would obtain  $\lambda \in \mathcal{L} = \{\{\}, \{BEGIN\}, \{END\}, \{BEGIN, END\}\}$ , which corresponds to  $\{BEGIN, END, UNIQUE, NEG\}$ . But for more than one annotation type, co-occurrence and implications can effectively be exploited and detected since these co-occurrences are explicitly used as training information. However, the granularity of this information is limited, since the approach is only able to generalize from the co-occurrence of two classes if there is no other class appearing since this would not generate the desired meta co-occurrence class anymore.

## 10.3 Evaluation

Since it is difficult to obtain densely annotated (free) corpora e.g. from the field of OBIE, we decided to generate our own dense dataset with the help of the Stanford Parser, which returns the syntactic structure of a sentence.<sup>68</sup> The result of the parser was considered to be the true and correct labeling of the corpus. The first six (scientific) texts from the *Learned* category of the Brown Corpus (Francis and Kucera 1979) were annotated with this tool, taking the first three documents for training and the remaining for testing. The resulting training set contains 6790 instances and 48 different annotations types, 7091 instances remained for testing. Since each annotation type leads to two tags denoting the

---

<sup>68</sup> <http://www-nlp.stanford.edu/software/lex-parser.shtml>

**Table 10.1:** Prediction quality of the different algorithms based on exact annotation matches, micro-averaged over all annotation types. MC for the traditional multiclass algorithm, ML for the multilabel transformation and LP for label powerset. LPC denotes the pairwise decomposition approach for multiclass problems.

Algorithm	Precision	Recall	F1
MC-BR	74.21	34.32	46.93
ML-BR	72.49	40.52	51.98
ML-QCLR	71.49	40.18	51.44
LP-BR	59.67	43.46	50.29
LP-LPC	65.12	41.73	50.87

*BEGIN* and the *END*, we obtain 96 different labels for the multilabel problem. In average there are 3.34 labels associated per token. For the label powerset representation, 334 classes were retrieved. A window size of 5 resulted in less than 7000 different features. We used the well known LibSVM library with linear kernel and standard settings as our base learner (Chang and Lin 2001).

The results are shown in Table 10.1 in terms of annotations-based micro-averaged recall, precision and F1 of exact matches (cf. Section 2.7) The first observation is that the multilabel approaches (QCLR and BR) seem to slightly outperform the traditional multiclass approach in terms of F1, and that ML has a slight advantage over LP. A closer look reveals that recall and precision highly depend on the used transformation approaches. LP seems to boost recall while ML and especially the classical multiclass approach improve precision, always at the expense of the opposite measure. The MC setting appears to generate quite conservative classifiers, since the MC extractor predicts 18% less annotations than ML-QCLR and even 28% less than LP-LPC.

Note that the absolute values may seem generally low, but remind that these results were produced without linguistic or any other intelligent preprocessing. Moreover, only exact annotation matches were taken into account, counting token matches improves the rates to around 70%.

## 10.4 Related Work

Only few attempts have been made on this subject so far. McDonald et al. (2005) were able to improve accuracy in extracting non-contiguous and overlapping segments of different types using an adapted multilabel algorithm. However, their algorithm is not directly comparable since it is centered and adapted to sentences whereas the approach presented here is token based and usable with any multilabel algorithm.

The independently developed approach of Finkel and Manning (2010) published approximately at the same time considers to jointly perform named entity recognition and

---

syntactic parsing. Their specialized conditional random field-based context-free grammar parser is able to outperform approaches which solve the tasks separately.

## 10.5 Summary

We have introduced the approach of presenting an information extraction problem as one multilabel classification problem rather than several independent multiclass problems. This view is more natural to the extraction problem and furthermore potentially allows the exploitation of relations and correlations between overlapping annotations.

Although all multilabel approaches achieve higher F1 scores in the experiments than the standard approach, a direct comparison of both approaches shows up to be difficult, since the traditional approach is focused on precision while the multilabel approaches obtained higher recall to the detriment of the precision. Evaluations on more corpora and perhaps with a more extensive also linguistic preprocessing are planned in order to obtain a clearer picture. Nevertheless, it has been demonstrated that the formulation as multilabel problem is at least comparable, particularly considering that the employed algorithms are not especially adapted or designed in order to exploit class correlations. Recent advantages in the exploitation of label dependencies let us expect substantial improvements. Promising advances were made e.g. in enhancing the pairwise approach by the detection and exploitation of present constraints on the labels, such as co-occurrences (Park and Fürnkranz 2008). The approach presented in Chapter 9 of incorporating label dependent features is surely another possibility.



---

## 11 Summary and Conclusions

Multilabel classification is becoming an increasingly important task in machine learning due to the growing number of application scenarios that require not only predicting one single top class as in multiclass classification, but a set or a ranking of relevant classes.

A prototypical application scenario for multilabel classification is the assignment of a set of keywords to a document, a frequently encountered problem in the text classification domain. With upcoming Web 2.0 technologies, this domain is extended by a wide range of tag suggestion tasks (cf. Section 2.9.1). This type of problem is often associated with a large number of instances, classes, or both, which demand for an efficient processing. The Reuters *rcv1* dataset for instance is composed of over 800,000 documents and 103 classes, a benchmark extracted from the *del.icio.us* platform contains almost 1000 classes and the introduced *EUR-Lex* database consists of almost 4000 classes (cf. Chapter 6). Moreover, the trend definitely is moving towards more data points and more labels.

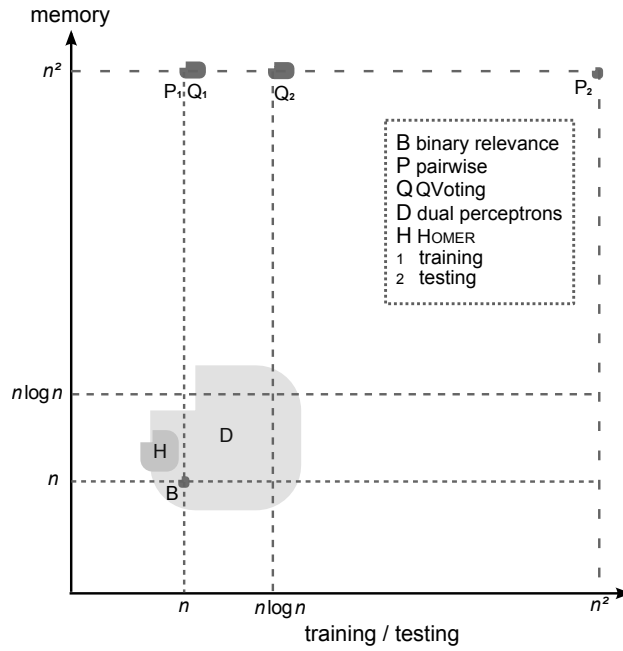
The work in this thesis was driven particularly by this explosive development. Hence, the emphasis was put on efficiency, though under the focused scenario this is directly connected to scalability. Effectivity, i.e. predictive quality, was ensured by the main pillar of this thesis: pairwise learning. This decomposition technique is based on the comparison of pairs of classes, which is the most fine-grained distinction we can make in a classification task. And it is exactly this characterizing property which gives pairwise learning the cutting edge over the competing approach, namely the predominant binary relevance learning.

But this property is simultaneously the most serious obstacle for pairwise learning, since the number of classifiers to be trained and the number of comparisons to be done for each test example grow quadratically with the number of classes. In light of our prototypical scenario, it was the main objective of this work to analyze and improve pairwise classification in face of the new challenging demands of current multilabel classification.

### 11.1 Challenges Revisited

As the present work demonstrates, this goal was not too ambitious. Several solutions were presented in order to manage the challenges initially presented in Section 1.1. The following section will revise these five challenges to efficient multilabel pairwise classification and the corresponding solutions that have been developed.

The basis for an efficient processing of a **high quantity of data** was set by the MLPP algorithm, which uses the fast yet effective perceptrons as base learners (Loza Mencía 2006). The **high number of features** does also not affect MLPP, since its efficiency depends on the feature density, which is generally low for textual data. The extension using *calibration* transformed the good ranker MLPP into the effective multilabel classi-



**Figure 11.1:** Schematic complexity diagram comparing the pairwise algorithms and binary relevance. Training and testing costs (x-axis, sub-indices 1 and 2 respectively are used if not equal) and memory requirements (y-axis) are shown with respect to the number of classes  $n$ . Areas indicate an additional dependency.

fier *CMLPP*. The fast processing speed of the incremental, anytime perceptrons ensures the suitability for **real-time applications** even on the 800,000 documents comprising the Reuters dataset.

However, the problem of the quadratic dependency on **number of classes** remained. The diagram in Figure 11.1 illustrates this: the starting point is a quadratic dependency  $n^2$  for the memory and testing ( $P_2$ ) and  $dn$  for training ( $P_1$ ). Note that training is decelerated by a factor equal to the (rather small) average labelset size  $d$  in comparison to the initial Figure 1.1. In consequence, the integration of the intelligent comparisons choosing scheme *QVoting* was a major milestone for multilabel pairwise classification. The log-linear scalability ( $Q_2$ ) is an important step towards binary relevance ( $B$ ) and enables *QCMLPP* and *QCLR* for a wide range of practical applications. However, the new large-scale *EUR-Lex* dataset with its almost 4000 classes was not amenable to this technique because of the high placement on the memory axis. The dual reformulation of the perceptrons allowed squeezing the 8 million classifiers into 1.4 instead of 152 GB. However, the *DMLPP* now depends linearly on the number of classes and training documents, which is reflected by the relatively large area in Figure 11.1 ( $D$ ).

The family of highly efficient and highly scalable multi-use algorithms for multilabel pairwise classification was completed by the combination with the hierarchical decomposition of HOMER (Tsoumakas et al. 2008). The relaxation of the pairwise setting allows

---

reducing the margin of  $H$ -QCLR to BR to a user-defined constant factor and it even reduces the training and testing costs dependence to sub-linearity.

This important advance prepared the ground for the connection to the highly related multi-task and multi-target learning, which was considered in this study for the first time (to the best of my knowledge). The investigation of *parallel tasks* was also the first attempt to analyze the exploitation of **label dependencies** in the pairwise setting. The use of multilabel learning algorithms for the application of automatically annotating text is closely related to the setting of the parallel tasks. Our work was the first time multilabel classification was applied in such a setting, with first positive advances. The possible connection with techniques for the exploration of local patterns was also explored in this work. *Subgroup discovery* based on exceptionality enables the further consideration of locally present label correlations.

## 11.2 Perspectives

A series of major and minor potential improvements were introduced throughout this work. For instance, the incorporation of more effective linear separation learners like SVMs instead of the basic perceptron, the incorporation of kernels (cf. Section 6.2.2), several simple solutions to the underestimation bias in the calibration (cf. Section 4.6.9), techniques for enabling the QVoting approach (cf. Section 5.5) and HOMER (cf. Section 7.4) to produce accurate rankings of labels in addition to labelset predictions, and many more.

However, for the author, the three main research avenues for efficient and effective pairwise classification in the multilabel setting are the following and they cover three main points: the further improvement of the efficiency and scalability with respect to large label spaces, the adjustment to the specific requirements of a task, and the enhanced exploitation of label dependencies.

The projection of the label output space to a much lower dimensional, substitute space (cf. Section 2.8.6) has the best chances, along with hierarchical transformations such as HOMER, of applying multilabel classification algorithms in general and pairwise decomposition in particular to problems with almost arbitrary label space size. The first track of this year's ECML/PKDD 2012 Discovery Challenge<sup>69</sup> demonstrates the need for this type of solutions: roughly 2.4 millions documents from the Wikipedia are to be mapped to approximately 325,000 categories. In order to solve this kind of problems, we are currently investigating techniques from semantic hashing and deep belief networks which in contrast to existing approaches do not require solving regression problems. Instead, conventional multilabel learning algorithms can be applied.

As the experimental evaluation in Section 9.6.3 has shown, the choice of the decomposition approach has a very diverse impact on the different multilabel evaluation measures. Hence, future multilabel learning algorithms should preferably be able to cover and ad-

---

<sup>69</sup> Third Challenge on Large Scale Hierarchical Text Classification <http://www.ecmlpkdd2012.net/info/discovery-challenge/>.

---

just to the diverse objectives and needs of specific multilabel tasks. In the pairwise framework, this can be achieved merely by changing the aggregation strategy of the pairwise predictions, particularly without touching the decomposition process and hence the pairwise ensemble. Two promising approaches were already proposed in Section 3.5.5, ranking through iterated choice and the posterior consideration of label constraints, which both deserve further investigation.

The third line of promising research concerns the extension of the pairwise preference learning framework by introducing currently ignored pairwise relations between labels. Notably, in the present framework, a base classifier dedicated to distinguish between two classes ignores the instances where both of these classes are simultaneously present (cf. Section 3.5.3). The inclusion of this additional information could further improve the ability of the pairwise approach to consider label dependencies. An extension in this direction affects the decomposition of the original problem as well as the aggregation of the base predictions. First theoretical reflections for a formal and semantically coherent realization in the pairwise preferences framework are promising, but considerably more effort still has to be invested.

### 11.3 Conclusions

In (almost) all experimental evaluations of the proposed solutions, we repeatedly obtained clear improvements in comparison to the use of binary relevance learning with respect to predictive quality. A thorough theoretical foundation was provided in order to substantiate these observations. It is the hope of the author to having contributed enough convincing evidence for the conceptional advantages of the pairwise approach.

Despite these important contributions, the main achievement of this work is the development of a set of approaches and techniques which enable the efficient and scalable application of the pairwise decomposition technique to large scale multilabel learning problems. This advancement allows a great variety of new applications which can benefit from the advantages of the pairwise approach.

---

# Bibliography

(Ailon and Mohri 2008)

Nir AILON, Mehryar MOHRI: *An Efficient Reduction of Ranking to Classification*. In: *21st Annual Conference on Learning Theory - COLT 2008, Helsinki, Finland, Proceedings*, eds. Rocco A. SERVEDIO, Tong ZHANG, pp. 87–98. Omnipress, 2008. <http://colt2008.cs.helsinki.fi/papers/32-Ailon.pdf>. (p. 77)

(Allwein et al. 2000)

Erin L. ALLWEIN, Robert E. SCHAPIRE, Yoram SINGER: *Reducing multiclass to binary: a unifying approach for margin classifiers*. In: *Journal of Machine Learning Research*, vol. 1: pp. 113–141, 2000. ISSN 1533-7928. (p. 58, 80, 81)

(Angulo et al. 2006)

Cecilio ANGULO, Francisco RUIZ, Luis GONZÁLEZ, Juan Antonio ORTEGA: *Multi-Classification by Using Tri-Class SVM*. In: *Neural Processing Letters*, vol. 23 (1): pp. 89–101, 2006. doi:10.1007/s11063-005-3500-3. (p. 16, 72)

(Bakker and Heskes 2003)

Bart BAKKER, Tom HESKES: *Task clustering and gating for bayesian multitask learning*. In: *Journal of Machine Learning Research*, vol. 4: pp. 83–99, 2003. <http://www.jmlr.org/papers/v4/bakker03a.html>. (p. 172)

(Banerjee and Ghosh 2006)

Arindam BANERJEE, Joydeep GHOSH: *Scalable Clustering Algorithms with Balancing Constraints*. In: *Data Mining and Knowledge Discovery*, vol. 13 (3): pp. 365–395, 2006. ISSN 1384-5810. (p. 156)

(Barnard et al. 2003)

Kobus BARNARD, Pinar DUYGULU, Nando DE FREITAS, David FORSYTH, David BLEI, Michael I. JORDAN: *Matching Words and Pictures*. In: *Journal of Machine Learning Research*, vol. 3: pp. 1107–1135, 2003. (p. 46)

(Bi and Kwok 2011)

Wei BI, James Tin-Yau KWOK: *Multi-Label Classification on Tree- and DAG-Structured Hierarchies*. In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, eds. Lise GETOOR, Tobias SCHEFFER, pp. 17–24. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0619-5. [http://www.icml-2011.org/papers/10\\_icmlpaper.pdf](http://www.icml-2011.org/papers/10_icmlpaper.pdf). (p. 41, 150)

---

Page number(s) in parenthesis at the end indicate the place(s) where each entry was referenced.

---

(Bian et al. 2011)

Guorui BIAN, Michael McALEER, Wing-Keung WONG: *A trinomial test for paired data when there are many ties*. In: *Mathematics and Computers in Simulation*, vol. 81 (6): pp. 1153 – 1160, 2011. ISSN 0378-4754. doi:10.1016/j.matcom.2010.11.002. <http://eprints.hkbu.edu.hk/287/1/WWK100.pdf>. (p. 49)

(Bishop 1995)

Christopher M. BISHOP: *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. ISBN 0198538642. (p. 85, 86, 87)

(Boutell et al. 2004)

Matthew R. BOUTELL, Jiebo LUO, Xipeng SHEN, C. M. Christopher M. BROWN: *Learning Multi-Label Scene Classification*. In: *Pattern Recognition*, vol. 37 (9): pp. 1757–1771, 2004. <http://www.rose-hulman.edu/~boutell/publications/boutell04PRmultilabel.pdf>. (p. 46, 54, 56, 57)

(Brinker and Hüllermeier 2006)

Klaus BRINKER, Eyke HÜLLERMEIER: *Case-Based Label Ranking*. In: *Proceedings of the 17th European Conference on Machine Learning (ECML-06)*, eds. Johannes FÜRNKRANZ, Tobias SCHEFFER, Myra SPILIOPOULOU, pp. 566–573. Springer-Verlag, Berlin, Germany, 2006. (p. 38)

(Brinker et al. 2006)

Klaus BRINKER, Johannes FÜRNKRANZ, Eyke HÜLLERMEIER: *A Unified Model for Multilabel Classification and Ranking*. In: *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI-06)*, eds. G. BREWKA, S. CORADESCHI, A. PERINI, P. TRAVERSO, pp. 489–493, 2006. <http://www.ke.tu-darmstadt.de/~juffi/publications/ecai-06.pdf>. (p. 7, 56, 61, 63, 64, 65, 66, 67, 84)

(Brucker et al. 2011)

Florian BRUCKER, Fernando BENITES, Elena SAPOZHNIKOVA: *An Empirical Comparison of Flat and Hierarchical Performance Measures for Multi-Label Classification with Hierarchy Extraction*. In: *Knowledge-Based and Intelligent Information and Engineering Systems*, vol. 6881 of *Lecture Notes in Computer Science*, eds. Andreas KÖNIG, Andreas DENGEL, Knut HINKELMANN, Koichi KISE, Robert HOWLETT, Lakhmi JAIN, pp. 579–589. Springer Berlin / Heidelberg, 2011a. ISBN 978-3-642-23850-5. doi:10.1007/978-3-642-23851-2\_59. [http://books.google.de/books?id=C1Fr\\_0KoESIC&pg=PA579](http://books.google.de/books?id=C1Fr_0KoESIC&pg=PA579). (p. 23, 35)

(Brucker et al. 2011)

Florian BRUCKER, Fernando BENITES, Elena SAPOZHNIKOVA: *Multi-label classification and extracting predicted class hierarchies*. In: *Pattern Recognition*, vol. 44 (3): pp. 724–738, 2011b. doi:10.1016/j.patcog.2010.09.010. <http://linkinghub.elsevier.com/retrieve/pii/S0031320310004504>. (p. 23)

(Burges 1998)

Christopher J. C. BURGESS: *A Tutorial on Support Vector Machines for Pattern Recognition*.

- 
- In: *Data Mining and Knowledge Discovery*, vol. 2 (2): pp. 121–167, 1998. <http://www.svms.org/tutorials/Burges1998.pdf>. (p. 87)
- (Cai and Hofmann 2004)  
Lijuan CAI, Thomas HOFMANN: *Hierarchical Document Categorization with Support Vector Machines*. In: *Proceedings of the 13-th ACM Conference on Information and Knowledge Management (CIKM-04)*, pp. 78–87. Washington, DC, 2004. (p. 22)
- (Caruana 1997)  
Rich CARUANA: *Multitask Learning*. In: *Machine Learning*, vol. 28: pp. 41–75, 1997. (p. 172)
- (Cesa-Bianchi et al. 2006)  
Nicolò CESA-BIANCHI, Claudio GENTILE, Luca ZANIBONI: *Incremental Algorithms for Hierarchical Classification*. In: *Journal of Machine Learning Research*, vol. 7: pp. 31–54, 2006. <http://www.jmlr.org/papers/v7/cesa-bianchi06a.html>. (p. 23, 93, 168)
- (Chang and Lin 2001)  
Chih-Chung CHANG, Chih-Jen LIN: *LIBSVM: a library for support vector machines*, 2001. Manual, Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>. (p. 2, 47, 79, 89, 124, 192, 207)
- (Chang and Lin 2012)  
Chih-Chung CHANG, Chih-Jen LIN: *LIBSVM Data: Multi-label Classification*. Website, 2012. <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html>. Last accessed at 2012-01-12. (p. 47, 109, 178)
- (Cheng and Hüllermeier 2009)  
Weiwei CHENG, Eyke HÜLLERMEIER: *Combining instance-based learning and logistic regression for multilabel classification*. In: *Machine Learning*, vol. 76 (2-3): pp. 211–225, 2009. doi:10.1007/s10994-009-5127-5. <http://www.mathematik.uni-marburg.de/~eyke/publications/ml09draft.pdf>. (p. 172, 199)
- (Cheng et al. 2010)  
Weiwei CHENG, Krzysztof DEMBCZYŃSKI, Eyke HÜLLERMEIER: *Graded multilabel classification: The ordinal case*. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, eds. Johannes FÜRNKRANZ, Thorsten JOACHIMS, pp. 223–230. Omnipress, Haifa, Israel, 2010. <http://www.icml2010.org/papers/596.pdf>. (p. 43)
- (Coakley and Heise 1996)  
Clint W. COAKLEY, Mark A. HEISE: *Versions of the sign test in the presence of ties*. In: *Biometrics*, vol. 52 (4): pp. 1242–1251, 1996. (p. 49)
- (Conover 1973)  
William Jay CONOVER: *On Methods of Handling Ties in the Wilcoxon Signed-Rank Test*. In: *Journal of the American Statistical Association*, vol. 68 (344): pp. 985–988, 1973. ISSN 01621459. <http://www.jstor.org/stable/2284536>. (p. 50)



---

(Cortes and Vapnik 1995)

Corinna CORTES, Vladimir VAPNIK: *Support-Vector Networks*. In: *Machine Learning*, vol. 20 (3): pp. 273–297, 1995. ISSN 0885-6125. doi:10.1007/BF00994018. [http://image.diku.dk/imagecanon/material/cortes\\_vapnik95.pdf](http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf). (p. 87)

(Crammer and Singer 2002)

Koby CRAMMER, Yoram SINGER: *A new family of online algorithms for category ranking*. In: *Proceedings of the 25th Annual International Conference on Research and Development in Information Retrieval (SIGIR 2002)*, pp. 151–158. ACM Press, New York, NY, USA, 2002. ISBN 1-58113-561-0. <http://www.cs.huji.ac.il/~singer/papers/mcml.ps.gz>. (p. 22, 38)

(Crammer and Singer 2003)

Koby CRAMMER, Yoram SINGER: *A Family of Additive Online Algorithms for Category Ranking*. In: *Journal of Machine Learning Research*, vol. 3 (6): pp. 1025–1058, 2003. <http://www.jmlr.org/papers/v3/crammer03b.html>. (p. 22, 32, 38, 63, 83, 90, 91, 97, 102, 172)

(Crammer et al. 2006)

Koby CRAMMER, Ofer DEKEL, Joseph KESHET, Shai SHALEV-SHWARTZ, Yoram SINGER: *Online Passive-Aggressive Algorithms*. In: *Journal of Machine Learning Research*, vol. 7: pp. 551–585, 2006. ISSN 1533-7928 (electronic); 1532-443. <http://jmlr.csail.mit.edu/papers/v7/crammer06a.html>. (p. 89, 150)

(de Melo and Siersdorfer 2007)

Gerard DE MELO, Stefan SIERSDORFER: *Multilingual Text Classification Using Ontologies*. In: *Advances in Information Retrieval, 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007, Proceedings*, vol. 4425 of *Lecture Notes in Computer Science*, eds. Giambattista AMATI, Claudio CARPINETO, Giovanni ROMANO, pp. 541–548. Springer, 2007. ISBN 978-3-540-71494-1. doi:10.1007/978-3-540-71496-5\_49. <http://www.l3s.de/~siersdorfer/sources/2007/ecir07siers.pdf>. (p. 129)

(Dekel et al. 2005)

Ofer DEKEL, Shai SHALEV-SHWARTZ, Yoram SINGER: *The Forgetron: A Kernel-Based Perceptron on a Fixed Budget*. In: *Advances in Neural Information Processing Systems 18*, 2005. [http://books.nips.cc/papers/files/nips18/NIPS2005\\_0192.pdf](http://books.nips.cc/papers/files/nips18/NIPS2005_0192.pdf). (p. 89, 104, 140)

(del Coz et al. 2009)

Juan José DEL COZ, Jorge DíEZ, Antonio BAHAMONDE: *Learning Nondeterministic Classifiers*. In: *Journal of Machine Learning Research*, vol. 10: pp. 2273–2293, 2009. ISSN 1532-4435. <http://jmlr.csail.mit.edu/papers/v10/delcoz09a.html>. (p. 18)

(Dembczyński et al. 2010)

Krzysztof DEMBCZYŃSKI, Weiwei CHENG, Eyke HÜLLERMEIER: *Bayes optimal multilabel classification via probabilistic classifier chains*. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, eds. Johannes FÜRNKRANZ, Thorsten



---

JOACHIMS, pp. 279–286. Omnipress, Haifa, Israel, 2010a. <http://www.mathematik.uni-marburg.de/~eyke/publications/589.pdf>. (p. 13, 22, 39, 40, 80, 172)

(Dembczyński et al. 2010)

Krzysztof DEMBCZYŃSKI, Willem WAEGEMAN, Weiwei CHENG, Eyke HÜLLERMEIER: *On label dependence in multi-label classification*. In: *Proceedings of the ICML-10 Workshop on Learning from Multi-Label Data*, eds. Min-Ling ZHANG, Grigorios TSOUMAKAS, Zhi-Hua ZHOU, pp. 5–12. Haifa, Israel, 2010b. <http://www.mathematik.uni-marburg.de/~eyke/publications/mld10.pdf>. (p. 23, 24)

(Dembczyński et al. 2010)

Krzysztof DEMBCZYŃSKI, Willem WAEGEMAN, Weiwei CHENG, Eyke HÜLLERMEIER: *Regret analysis for performance metrics in multi-label classification: the case of hamming and subset zero-one loss*. In: *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD'10): Part I*, pp. 280–295. Springer-Verlag, Berlin, Heidelberg, 2010c. ISBN 3-642-15879-X, 978-3-642-15879-7. [http://www.mathematik.uni-marburg.de/~eyke/publications/dembczynski\\_762.pdf](http://www.mathematik.uni-marburg.de/~eyke/publications/dembczynski_762.pdf). (p. 13, 22, 36, 76)

(Demšar 2006)

Janez DEMŠAR: *Statistical Comparisons of Classifiers over Multiple Data Sets*. In: *Journal of Machine Learning Research*, vol. 7: pp. 1–30, 2006. <http://jmlr.csail.mit.edu/papers/v7/demsar06a.html>. (p. 48, 49, 50, 51)

(Demšar 2008)

Janez DEMŠAR: *On the Appropriateness of Statistical Tests in Machine Learning*. In: *ICML 2008 Third Workshop on Evaluation Methods for Machine Learning*, 2008. [http://www.site.uottawa.ca/ICML08WS/papers/J\\_Demsar.pdf](http://www.site.uottawa.ca/ICML08WS/papers/J_Demsar.pdf). (p. 48)

(Dietterich and Bakiri 1995)

Thomas G. DIETTERICH, Ghulum BAKIRI: *Solving Multiclass Learning Problems via Error-Correcting Output Codes*. In: *Journal of Artificial Intelligence Research*, vol. 2: pp. 263–286, 1995. (p. 58, 78)

(Diplaris et al. 2005)

Sotiris DIPLARIS, Grigorios TSOUMAKAS, Pericles A. MITKAS, Ioannis P. VLAHAVAS: *Protein Classification with Multiple Algorithms*. In: *Advances in Informatics, 10th Panhellenic Conference on Informatics, PCI 2005, Proceedings*, vol. 3746 of *Lecture Notes in Computer Science*, eds. Panayiotis BOZANIS, Elias N. HOUSTIS, pp. 448–456. Springer, 2005. ISBN 3-540-29673-5. doi:10.1007/11573036\_42. <http://lpis.csd.auth.gr/publications/tsoumakas-pci2005a.pdf>. (p. 47)

(Domingos 1997)

Pedro DOMINGOS: *A Unified Approach to Concept Learning*. Dissertation, University of California, Irvine, 1997. <http://www.cs.washington.edu/homes/pedrod/papers/phd.pdf>. (p. 15)

---

(Duivesteijn et al. 2010)

Wouter DUIVESTEIJN, Arno J. KNOBBE, Ad FEELDERS, Matthijs VAN LEEUWEN: *Subgroup Discovery Meets Bayesian Networks – An Exceptional Model Mining Approach*. In: *Proceedings of the 2010 IEEE International Conference on Data Mining, ICDM '10*, pp. 158–167. IEEE Computer Society, Washington, DC, USA, 2010. ISBN 978-0-7695-4256-0. doi:10.1109/icdm.2010.53. <http://www.kiminkii.com/publications/icdm2010.pdf>. (p. 183, 185, 187, 189)

(Duivesteijn et al. 2012)

Wouter DUIVESTEIJN, Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ, Arno J. KNOBBE: *Multi-label LeGo – Enhancing Multi-label Classifiers with Local Patterns*. In: *Advances in Intelligent Data Analysis XI – Proceedings of the 11th International Symposium on Data Analysis (IDA-11)*, vol. 7619 of *Lecture Notes in Computer Science*, eds. Jaakko HOLLMÉN, Frank KLAUONN, Allan TUCKER, pp. 114–125. Springer, 2012. ISBN 978-3-642-34155-7. doi:10.1007/978-3-642-34156-4\_12. <http://www.ke.tu-darmstadt.de/publications/papers/IDA-12.pdf>. (p. 8, 199)

(Duygulu et al. 2002)

Pinar DUYGULU, Kobus BARNARD, Nando DE FREITAS, David FORSYTH: *Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary*. In: *ECCV '02 Proceedings of the 7th European Conference on Computer Vision-Part IV*, pp. 97–112. Springer-Verlag, 2002. ISBN 3-540-43748-7. (p. 46)

(Elisseeff and Weston 2001)

André ELISSEEFF, Jason WESTON: *A Kernel Method for Multi-Labelled Classification*. In: *Advances in Neural Information Processing Systems*, vol. 14, eds. Thomas G. DIETTERICH, Suzanna BECKER, Zoubin GHAHRAMANI, pp. 681–687. MIT Press, 2001. <http://books.nips.cc/papers/files/nips14/AA45.pdf>. (p. 22, 37, 47, 56, 63, 74, 82, 172)

(Evgeniou et al. 2006)

T. EVGENIOU, C. A. MICCHELLI, M. PONTIL: *Learning multiple tasks with kernel methods*. In: *Journal of Machine Learning Research*, vol. 6 (1): pp. 615–637, 2006. <http://www.jmlr.org/papers/v6/evgeniou05a.html>. (p. 172)

(Fan and Lin 2007)

Rong-En FAN, Chih-Jen LIN: *A Study on Threshold Selection for Multi-label Classification*. Tech. rep., National Taiwan University, 2007. <http://www.csie.ntu.edu.tw/~cjlin/papers/threshold.pdf>. (p. 74, 198)

(Fan et al. 2008)

Rong-En FAN, Kai-Wei CHANG, Cho-Jui HSIEH, Xiang-Rui WANG, Chih-Jen LIN: *LIBLINEAR: A Library for Large Linear Classification*. In: *Journal of Machine Learning Research*, vol. 9: pp. 1871–1874, 2008. ISSN 1533-7928 (electronic); 1532-443. (p. 89, 124, 181, 192)

---

(Finkel and Manning 2010)

Jenny Rose FINKEL, Christopher D. MANNING: *Hierarchical joint learning: improving joint parsing and named entity recognition with non-jointly labeled data*. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pp. 720–728. Association for Computational Linguistics, Stroudsburg, PA, USA, 2010. <http://nlp.stanford.edu/jrfinkel/papers/hier-joint.pdf>. (p. 172, 207)

(Fong et al. 2003)

Daniel Y. T. FONG, C. W. KWAN, K. F. LAM, Karen S. L. LAM: *Use of the Sign Test for the Median in the Presence of Ties*. In: *The American Statistician*, vol. 57 (4): pp. 237–240, 2003. ISSN 00031305. <http://www.jstor.org/stable/30037288>. (p. 49)

(Francis and Kucera 1979)

W. N. FRANCIS, H. KUCERA: *Brown Corpus Manual*. Tech. rep., Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979. <http://icame.uib.no/brown/bcm.html>. (p. 206)

(Frank and Asuncion 2010)

A. FRANK, A. ASUNCION: *UCI Machine Learning Repository*, 2010. <http://archive.ics.uci.edu/ml>. Dataset Repository. (p. 17, 71, 78)

(Freund and Schapire 1999)

Yoav FREUND, Robert E. SCHAPIRE: *Large Margin Classification Using the Perceptron Algorithm*. In: *Machine Learning*, vol. 37 (3): pp. 277–296, 1999. <http://www.cse.ucsd.edu/~yfreund/papers/LargeMarginsUsingPerceptron.pdf>. (p. 87, 89, 92)

(Friedman 1937)

Milton FRIEDMAN: *The use of ranks to avoid the assumption of normality implicit in the analysis of variance*. In: *Journal of the American Statistical Association*, vol. 32: pp. 675–701, 1937. (p. 50)

(Friedman 1940)

Milton FRIEDMAN: *A comparison of alternative tests of significance for the problem of m rankings*. In: *Annals of Mathematical Statistics*, vol. 11: pp. 86–92, 1940. (p. 50)

(Fürnkranz 2002)

Johannes FÜRNKRANZ: *Round Robin Classification*. In: *Journal of Machine Learning Research*, vol. 2: pp. 721–747, 2002. <http://www.ai.mit.edu/projects/jmlr/papers/volume2/fuernkranz02a/html/>. (p. 2, 57, 59, 61, 66, 67, 70, 71, 79)

(Fürnkranz and Hüllermeier 2003)

Johannes FÜRNKRANZ, Eyke HÜLLERMEIER: *Pairwise Preference Learning and Ranking*. In: *Proceedings of the 14th European Conference on Machine Learning (ECML-03)*, vol. 2837 of *Lecture Notes in Artificial Intelligence*, eds. Nada LAVRAČ, D. GAMBERGER, Hendrik BLOCKEEL, L. TODOROVSKI, pp. 145–156. Springer-Verlag, Cavtat, Croatia, 2003. <http://www.ke.tu-darmstadt.de/~juffi/publications/ecml-03.pdf>. (p. 60)

---

(Fürnkranz and Hüllermeier 2010)

Johannes FÜRNKRANZ, Eyke HÜLLERMEIER (eds.): *Preference Learning*. Springer-Verlag, 2010. ISBN 978-3642141249. doi:10.1007/978-3-642-14125-6. <http://www.springer.com/978-3-642-14124-9>. (p. 19)

(Fürnkranz and Knobbe 2010)

Johannes FÜRNKRANZ, Arno J. KNOBBE: *Special Issue on Global Modeling using Local Patterns*. In: *Data Mining and Knowledge Discovery*, vol. 12 (5), 2010. ISSN 1384-5810. (p. 183, 184, 185)

(Fürnkranz and Sima 2010)

Johannes FÜRNKRANZ, Jan Frederik SIMA: *On Exploiting Hierarchical Label Structure with Pairwise Classifiers*. In: *SIGKDD Explorations*, vol. 12 (2): pp. 21–25, 2010. <http://www.sigkdd.org/explorations/issues/12-2-2010-12/v12-02-6-UR-Fuernkranz.pdf>. Special Issue on Mining Unexpected Results. (p. 23)

(Fürnkranz et al. 2008)

Johannes FÜRNKRANZ, Eyke HÜLLERMEIER, Eneldo LOZA MENCÍA, Klaus BRINKER: *Multilabel Classification via Calibrated Label Ranking*. In: *Machine Learning*, vol. 73 (2): pp. 133–153, 2008. doi:10.1007/s10994-008-5064-8. <http://www.ke.tu-darmstadt.de/publications/papers/MLJ08.pdf>. (p. 7, 33, 34, 63, 64, 65, 66, 67, 70, 84, 105)

(Fürnkranz et al. 2012)

Johannes FÜRNKRANZ, Dragan GAMBERGER, Nada LAVRAČ: *Foundations of Rule Learning*. Springer-Verlag, 2012. ISBN 978-3-540-75196-0. doi:10.1007/978-3-540-75197-7. <http://www.springer.com/978-3-540-75196-0>. (p. 16, 184)

(Galar et al. 2011)

Mikel GALAR, Alberto FERNÁNDEZ, Edurne BARRENECHEA, Humberto BUSTINCE, Francisco HERRERA: *An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes*. In: *Pattern Recognition*, vol. 44 (8): pp. 1761–1776, 2011. ISSN 0031-3203. doi:10.1016/j.patcog.2011.01.017. <http://sci2s.ugr.es/publications/ficheros/2011-pr-galar-ovo-ova.pdf>. (p. 62, 77, 78, 79)

(García and Herrera 2008)

Salvador GARCÍA, Francisco HERRERA: *An Extension on “Statistical Comparisons of Classifiers over Multiple Data Sets” for all Pairwise Comparisons*. In: *Journal of Machine Learning Research*, vol. 9: pp. 2677–2694, 2008. ISSN 1533-7928. <http://www.jmlr.org/papers/volume9/garcia08a/garcia08a.pdf>. (p. 51)

(Ghamrawi and McCallum 2005)

Nadia GHAMRAWI, Andrew Kachites MCCALLUM: *Collective multi-label classification*. In: *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 195–200. ACM, New York, NY, USA, 2005. ISBN 1-59593-140-6. doi:10.1145/1099554.1099591. (p. 30, 172)

---

(Godbole and Sarawagi 2004)

Shantanu GODBOLE, Sunita SARAWAGI: *Discriminative Methods for Multi-labeled Classification*. In: *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004, Proceedings*, vol. 3056 of *Lecture Notes in Computer Science*, eds. Honghua DAI, Ramakrishnan SRIKANT, Chengqi ZHANG, pp. 22–30. Springer, 2004. ISBN 3-540-22064-X. doi:10.1007/978-3-540-24775-3\_5. (p. 200)

(Hayes and Weinstein 1991)

Philip J. HAYES, Steven P. WEINSTEIN: *CONSTRUE/TIS: A System for Content-Based Indexing of a Database of News Stories*. In: *Proceedings of the 2nd Conference on Innovative Applications of Artificial Intelligence (IAAI-90), May 1-3, 1990, Washington, DC, USA*, eds. Alain T. RAPPAPORT, Reid G. SMITH, IAAI '90, pp. 49–64. AAAI Press, Chicago, IL, USA, 1991. ISBN 0-262-68068-8. <http://aaai.org/Papers/IAAI/1990/IAAI90-006.pdf>. (p. 4, 56)

(Hsu and Lin 2002)

Chih-Wei HSU, Chih-Jen LIN: *A Comparison of Methods for Multi-class Support Vector Machines*. In: *IEEE Transactions on Neural Networks*, vol. 13 (2): pp. 415–425, 2002. <http://www.csie.ntu.edu.tw/~cjlin/papers/multisvm.pdf>. (p. 71, 79)

(Hsu et al. 2009)

Chih-Wei HSU, Chih-Chung CHANG, Chih-Jen LIN: *A Practical Guide to Support Vector Classification*. Tech. rep., Department of Computer Science, National Taiwan University, 2009a. Available at <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>. (p. 124)

(Hsu et al. 2009)

Daniel HSU, Sham KAKADE, John LANGFORD, Tong ZHANG: *Multi-Label Prediction via Compressed Sensing*. In: *Advances in Neural Information Processing Systems 22*, pp. 772–780. Curran Associates, 2009b. (p. 41, 150)

(Hühn and Hüllermeier 2009)

Jens C. HÜHN, Eyke HÜLLERMEIER: *FR3: A Fuzzy Rule Learner for Inducing Reliable Classifiers*. In: *IEEE Transactions on Fuzzy Systems*, vol. 17 (1): pp. 138–149, 2009. [http://www.mathematik.uni-marburg.de/~eyke/publications/fr3\\_draft.pdf](http://www.mathematik.uni-marburg.de/~eyke/publications/fr3_draft.pdf). (p. 20, 78)

(Hüllermeier and Brinker 2008)

Eyke HÜLLERMEIER, Klaus BRINKER: *Learning Valued Preference Structures for Solving Classification Problems*. In: *Fuzzy Sets and Systems*, vol. 159 (18): pp. 2337–2352, 2008. <http://www.mathematik.uni-marburg.de/~eyke/publications/fss08.pdf>. (p. 20, 78)

(Hüllermeier and Fürnkranz 2007)

Eyke HÜLLERMEIER, Johannes FÜRNKRANZ: *On Minimizing the Position Error in Label Ranking*. In: *Proceedings of 18th European Conference on Machine Learning (ECML-07)*,

- 
- eds. J. N. KOK, J. KORONACKI, Ramon LÓPEZ DE MÁNTARAS, S. MATWIN, Dunja MLADENIĆ, A. SKOWRON, pp. 583–590. Springer-Verlag, Warsaw, Poland, 2007. <http://www.ke.tu-darmstadt.de/~juffi/publications/ecml-07-PositionError.pdf>. (p. 76, 201)
- (Hüllermeier and Vanderlooy 2010)  
Eyke HÜLLERMEIER, Stijn VANDERLOOY: *Combining predictions in pairwise classification: An optimal adaptive voting strategy and its relation to weighted voting*. In: *Pattern Recognition*, vol. 43 (1): pp. 128–142, 2010. ISSN 0031-3203. doi:10.1016/j.patcog.2009.06.013. <http://www.mathematik.uni-marburg.de/~eyke/publications/PR09draft.pdf>. (p. 62, 77, 78)
- (Hüllermeier et al. 2008)  
Eyke HÜLLERMEIER, Johannes FÜRNKRANZ, Weiwei CHENG, Klaus BRINKER: *Label Ranking by Learning Pairwise Preferences*. In: *Artificial Intelligence*, vol. 172: pp. 1897–1916, 2008. doi:10.1016/j.artint.2008.08.002. (p. 19, 60, 77)
- (Ioannou et al. 2010)  
Marios IOANNOU, George SAKKAS, Grigorios TSOUMAKAS, Ioannis P. VLAHAVAS: *Obtaining Bipartitions from Score Vectors for Multi-Label Classification*. In: *Proceedings of the 2010 22nd IEEE International Conference on Tools with Artificial Intelligence - Volume 01, ICTAI '10*, pp. 409–416. IEEE Computer Society, Washington, DC, USA, 2010. ISBN 978-0-7695-4263-8. doi:10.1109/ictai.2010.65. (p. 74)
- (Joachims 1998)  
Thorsten JOACHIMS: *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. In: *Proceedings of 10th European Conference on Machine Learning (ECML-98)*, eds. C. NÉDELLEC, C. ROUVEIROL, pp. 137–142. Springer-Verlag, Chemnitz, Germany, 1998. <http://www.cs.iastate.edu/~jtian/cs573/Papers/Joachims-ECML-98.pdf>. (p. 56, 88)
- (Joachims 2002)  
Thorsten JOACHIMS: *Optimizing Search Engines Using Clickthrough Data*. In: *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-02)*, pp. 133–142. ACM Press, 2002. [http://www.cs.cornell.edu/People/tj/publications/joachims\\_02c.pdf](http://www.cs.cornell.edu/People/tj/publications/joachims_02c.pdf). (p. 82)
- (Joachims 2006)  
Thorsten JOACHIMS: *Training linear SVMs in linear time*. In: *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, eds. Tina ELIASSI-RAD, Lyle H. UNGAR, Mark CRAVEN, Dimitrios GUNOPULOS, pp. 217–226. ACM, 2006. ISBN 1-59593-339-5. doi:10.1145/1150402.1150429. (p. 82, 89)
- (Katakis et al. 2008)  
Ioannis KATAKIS, Grigorios TSOUMAKAS, Ioannis P. VLAHAVAS: *Multilabel Text Classification for Automated Tag Suggestion*. In: *Proceedings of the ECML/PKDD-08 Workshop*



---

on Discovery Challenge. Antwerp, Belgium, 2008. [http://lpis.csd.auth.gr/publications/katakis\\_ecmlpkdd08\\_challenge.pdf](http://lpis.csd.auth.gr/publications/katakis_ecmlpkdd08_challenge.pdf). (p. 43, 46)

(Kazawa et al. 2005)

Hideto KAZAWA, Tomonori IZUMITANI, Hirotoshi TAIRA, Eisaku MAEDA: *Maximal Margin Labeling for Multi-Topic Text Categorization*. In: *Advances in Neural Information Processing Systems 17*, eds. Lawrence K. SAUL, Yair WEISS, Léon BOTTOU, pp. 649–656. MIT Press, Cambridge, MA, 2005. [http://books.nips.cc/papers/files/nips17/NIPS2004\\_0238.pdf](http://books.nips.cc/papers/files/nips17/NIPS2004_0238.pdf). (p. 40)

(Khardon and Wachman 2007)

Roni KHARDON, Gabriel WACHMAN: *Noise Tolerant Variants of the Perceptron Algorithm*. In: *Journal of Machine Learning Research*, vol. 8: pp. 227–248, 2007. ISSN 1533-7928 (electronic); 1532-4443. <http://jmlr.csail.mit.edu/papers/v8/khardon07a.html>. (p. 89)

(Knerr et al. 1990)

Stefan KNERR, Léon PERSONNAZ, Gérard DREYFUS: *Single-Layer Learning Revisited: A Stepwise Procedure for Building and Training a Neural Network*. In: *Neurocomputing: Algorithms, Architectures and Applications*, vol. F68 of NATO ASI Series, eds. F. FOGELMAN SOULIÉ, J. HÉRAULT, pp. 41–50. Springer-Verlag, 1990. (p. 2, 79)

(Knerr et al. 1992)

Stefan KNERR, Léon PERSONNAZ, Gérard DREYFUS: *Handwritten Digit Recognition by Neural Networks with Single-Layer Training*. In: *IEEE Transactions on Neural Networks*, vol. 3 (6): pp. 962–968, 1992. (p. 71, 92)

(Knobbe and Valkonet 2009)

Arno J. KNOBBE, Joris VALKONET: *Building Classifiers from Pattern Teams*. In: *From Local Patterns to Global Models: Proceedings of the ECML/PKDD-09 Workshop*, 2009. <http://www.ke.tu-darmstadt.de/events/LeGo-09/08-Knobbe.pdf>. (p. 185)

(Knobbe et al. 2008)

Arno J. KNOBBE, Bruno CRÉMILLEUX, Johannes FÜRNKRANZ, Martin SCHOLZ: *From Local Patterns to Global Models: The LeGo Approach to Data Mining*. In: *From Local Patterns to Global Models: Proceedings of the ECML/PKDD-08 Workshop (LeGo-08)*, ed. Arno J. KNOBBE, pp. 1–16. Antwerp, Belgium, 2008. <http://www.ke.tu-darmstadt.de/events/LeGo-08/1.pdf>. (p. 183, 184)

(Koller and Sahami 1997)

Daphne KOLLER, Mehran SAHAMI: *Hierarchically Classifying Documents Using Very Few Words*. In: *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, pp. 170–178. Nashville, 1997. (p. 156)

(Kong and Yu 2011)

Xiangnan KONG, Philip YU: *gMLC: a multi-label feature selection framework for graph classification*. In: *Knowledge and Information Systems*, pp. 1–25, 2011a. ISSN 0219-1377. doi:10.1007/s10115-011-0407-3. (p. 41)

---

(Kong and Yu 2011)

Xiangnan KONG, Philip S. YU: *An ensemble-based approach to fast classification of multi-label data streams*. In: *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2011*, eds. Dimitrios GEORGAKOPOULOS, James B. D. JOSHI, pp. 95–104. IEEE, 2011b. ISBN 978-1-4673-0683-6. <http://www.cs.uic.edu/~xkong/mlstream.pdf>. (p. 38, 103, 104)

(Kong et al. 2011)

Xiangnan KONG, Xiaoxiao SHI, Philip S. YU: *Multi-Label Collective Classification*. In: *Proceedings of the Eleventh SIAM International Conference on Data Mining, SDM 2011, April 28-30, 2011, Mesa, Arizona, USA*, pp. 618–629. SIAM / Omnipress, 2011. ISBN 978-0-898719-92-5. <http://siam.omnibooksonline.com/2011datamining/data/papers/001.pdf#page=1>. (p. 41)

(Kotłowski et al. 2011)

Wojciech KOTŁOWSKI, Krzysztof DEMBCZYŃSKI, Eyke HÜLLERMEIER: *Bipartite Ranking through Minimization of Univariate Loss*. In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, eds. Lise GETOOR, Tobias SCHEFFER, ICML '11, pp. 1113–1120. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0619-5. <http://www.mathematik.uni-marburg.de/~eyke/publications/icml11.pdf>. (p. 32, 76)

(Krönke 2011)

Tobias KRÖNKE: *Verbesserungen des Perzeptron-Algorithmus*. Master's thesis, TU Darmstadt, Knowledge Engineering Group, 2011. [http://www.ke.tu-darmstadt.de/lehre/arbeiten/bachelor/2011/Kroenke\\_Tobias.pdf](http://www.ke.tu-darmstadt.de/lehre/arbeiten/bachelor/2011/Kroenke_Tobias.pdf). Software available at <http://www.ke.tu-darmstadt.de/resources/perceptrovement>. (p. 89)

(Kuncheva 2004)

Ludmila I. KUNCHEVA: *Combining Pattern Classifiers : Methods and Algorithms*. Wiley-Interscience, 2004. ISBN 0471210781. <http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471210781.html>. (p. 61, 77)

(Leeuwen 2010)

Matthijs LEEUWEN: *Maximal exceptions with minimal descriptions*. In: *Data Mining and Knowledge Discovery*, vol. 21: pp. 259–276, 2010. ISSN 1384-5810. doi:10.1007/s10618-010-0187-5. (p. 183, 185)

(Leman et al. 2008)

Dennis LEMAN, Ad FEELDERS, Arno J. KNOBBE: *Exceptional Model Mining*. In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD-2008), Part II*, pp. 1–16, 2008. doi:10.1007/978-3-540-87481-2\_1. (p. 183, 185)

(Lewis 1992)

David Dolan LEWIS: *An Evaluation of Phrasal and Clustered Representations on a Text Cat-*



---

egorization Task. In: *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 37–50, 1992. (p. 4)

(Lewis 2004)

David Dolan LEWIS: *Reuters-21578 Text Categorization Test Collection Distribution 1.0*. README file (V 1.3), 2004. <http://www.daviddlewis.com/resources/testcollections/reuters21578/>. (p. 4, 45, 47)

(Lewis et al. 2004)

David Dolan LEWIS, Yiming YANG, Tony G. ROSE, Fan LI: *RCV1: A New Benchmark Collection for Text Categorization Research*. In: *Journal of Machine Learning Research*, vol. 5: pp. 361–397, 2004. <http://jmlr.csail.mit.edu/papers/v5/lewis04a.html>. (p. 4, 29, 45, 47, 83)

(Li and Bontcheva 2007)

Yaoyong LI, Kalina BONTCHEVA: *Hierarchical, perceptron-like learning for ontology-based information extraction*. In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, eds. Carey L. WILLIAMSON, Mary Ellen ZURKO, Peter F. PATEL-SCHNEIDER, Prashant J. SHENOY, pp. 777–786. ACM, 2007. ISBN 978-1-59593-654-7. doi:10.1145/1242572.1242677. <http://www2007.org/papers/paper428.pdf>. (p. 93, 203)

(Li et al. 2002)

Yaoyong LI, Hugo ZARAGOZA, Ralf HERBRICH, John SHAWE-TAYLOR, Jaz S. KANDOLA: *The Perceptron Algorithm with Uneven Margins*. In: *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002)*, eds. Claude SAMMUT, Achim G. HOFFMANN, pp. 379–386. Morgan Kaufmann, 2002. ISBN 1-55860-873-7. (p. 89, 93, 150)

(Lin and Lin 2003)

Hsuan-tien LIN, Chih-Jen LIN: *A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods*. Tech. rep., Department of Computer Science, National Taiwan University, 2003. <http://www.csie.ntu.edu.tw/~cjlin/papers/tanh.pdf>. (p. 77)

(Loza Mencía 2006)

Eneldo LOZA MENCÍA: *Paarweises Lernen von Multilabel-Klassifikationen mit dem Perzeptron-Algorithmus*. Diploma thesis, TU Darmstadt, Knowledge Engineering Group, 2006. [http://www.ke.tu-darmstadt.de/lehre/arbeiten/diplom/2006/Loza\\_Eneldo.pdf](http://www.ke.tu-darmstadt.de/lehre/arbeiten/diplom/2006/Loza_Eneldo.pdf). (p. 7, 33, 45, 71, 77, 84, 209)

(Loza Mencía 2009)

Eneldo LOZA MENCÍA: *Segmentation of legal documents*. In: *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pp. 88–97. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-597-0. doi:10.1145/1568234.1568245. <http://www.ke.tu-darmstadt.de/publications/papers/loza09segmentation.pdf>. (p. 204)

---

(Loza Mencía 2010)

Eneldo LOZA MENCÍA: *An Evaluation of Multilabel Classification for the Automatic Annotation of Texts*. In: *Proceedings of the LWA 2010: Lernen, Wissen, Adaptivität, Workshop on Knowledge Discovery, Data Mining and Machine Learning (KDML 2010)*, eds. Martin ATZMÜLLER, Dominik BENZ, Andreas HOTH, Gerd STUMME, pp. 121–123, 2010a. <http://www.kde.cs.uni-kassel.de/conf/lwa10/papers/kdml11.pdf>. (p. 9)

(Loza Mencía 2010)

Eneldo LOZA MENCÍA: *Multilabel Classification in Parallel Tasks*. In: Zhang et al. (2010a), pp. 29–36. <http://www.ke.tu-darmstadt.de/publications/papers/loza10mlpt.pdf>. (p. 8)

(Loza Mencía and Fürnkranz 2007)

Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ: *An Evaluation of Efficient Multilabel Classification Algorithms for Large-Scale Problems in the Legal Domain*. In: *Proceedings of the LWA 2007: Lernen - Wissen - Adaption*, ed. Alexander HINNEBURG, pp. 126–132. Halle, Germany, 2007a. ISBN 978-3-86010-907-6. <http://www.ke.tu-darmstadt.de/publications/papers/LWA07.pdf>. (p. 8, 45, 46, 148)

(Loza Mencía and Fürnkranz 2007)

Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ: *Pairwise Learning of Multilabel Classifications with Perceptrons*. Tech. Rep. TUD-KE-2007-05, TU Darmstadt, Knowledge Engineering Group, 2007b. <http://www.ke.tu-darmstadt.de/publications/reports/tud-ke-2007-05.pdf>. (p. 84, 105)

(Loza Mencía and Fürnkranz 2008)

Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ: *Efficient Pairwise Multilabel Classification for Large-Scale Problems in the Legal Domain*. In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD-2008), Part II*, vol. 5212 of *Lecture Notes in Computer Science*, eds. Walter DAELEMANS, Bart GOETHALS, Katharina MORIK, pp. 50–65. Springer, Antwerp, Belgium, 2008a. ISBN 978-3-540-87480-5. doi:10.1007/978-3-540-87481-2\_4. <http://www.ke.tu-darmstadt.de/publications/papers/ECML08.pdf>. Accompanying EUR-Lex dataset available at <http://www.ke.tu-darmstadt.de/resources/eurlex>. (p. 8, 132, 143, 151)

(Loza Mencía and Fürnkranz 2008)

Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ: *An Evaluation of Efficient Multilabel Classification Algorithms for Large-Scale Problems in the Legal Domain*. In: *Proceedings of the LREC 2008 Workshop on Semantic Processing of Legal Texts*, eds. Simonetta MONTEMAGNI, Daniela TISCORNIA, Enrico FRANCESCONI, Wim PETERS, pp. 23–32. Marrakech, Morocco, 2008b. <http://www.ke.tu-darmstadt.de/publications/papers/LREC2008.pdf>. (p. 132, 143)

(Loza Mencía and Fürnkranz 2008)

Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ: *Pairwise Learning of Multilabel Classifica-*

---

tions with Perceptrons. In: *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IJCNN-08)*, pp. 2900–2907. IEEE, Hong Kong, 2008c. ISBN 978-1-4244-1821-3. doi:10.1109/IJCNN.2008.4634206. (p. 7, 84, 105)

(Loza Mencía and Fürnkranz 2010)

Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ: *Efficient Multilabel Classification Algorithms for Large-Scale Problems in the Legal Domain*. In: *Semantic Processing of Legal Texts – Where the Language of Law Meets the Law of Language*, vol. 6036 of *Lecture Notes in Artificial Intelligence*, eds. Enrico FRANCESCONI, Simonetta MONTEMAGNI, Wim PETERS, Daniela TISCORNIA, 1 ed., pp. 192–215. Springer-Verlag, 2010. ISBN 978-3-642-12836-3. doi:10.1007/978-3-642-12837-0\_11. <http://www.ke.tu-darmstadt.de/publications/papers/loza10eurlex.pdf>. Accompanying EUR-Lex dataset available at <http://www.ke.tu-darmstadt.de/resources/eurlex>. (p. 8, 47, 107)

(Loza Mencía et al. 2008)

Eneldo LOZA MENCÍA, Sang-Hyeun PARK, Johannes FÜRNKRANZ: *Advances in Efficient Pairwise Multilabel Classification*. Tech. Rep. TUD-KE-2008-06, TU Darmstadt, Knowledge Engineering Group, 2008. <http://www.ke.tu-darmstadt.de/publications/reports/tud-ke-2008-06.pdf>. (p. 123)

(Loza Mencía et al. 2009)

Eneldo LOZA MENCÍA, Sang-Hyeun PARK, Johannes FÜRNKRANZ: *Efficient Voting Prediction for Pairwise Multilabel Classification*. In: *Proceedings of the 17th European Symposium on Artificial Neural Networks (ESANN 2009, Bruges, Belgium)*, pp. 117–122. d-side publications, 2009. ISBN 2-930307-09-9. <http://www.dice.ucl.ac.be/Proceedings/esann/esannpdf/es2009-112.pdf>. (p. 7)

(Loza Mencía et al. 2010)

Eneldo LOZA MENCÍA, Sang-Hyeun PARK, Johannes FÜRNKRANZ: *Efficient Voting Prediction for Pairwise Multilabel Classification*. In: *Neurocomputing*, vol. 73 (7-9): pp. 1164 – 1176, 2010. ISSN 0925-2312. doi:10.1016/j.neucom.2009.11.024. <http://www.ke.tu-darmstadt.de/publications/papers/neucom10.pdf>. (p. 7, 107, 123)

(Madjarov et al. 2012)

Gjorgji MADJAROV, Dejan GJORGJEVIKJ, Sašo DŽEROSKI: *Two stage architecture for multi-label learning*. In: *Pattern Recognition*, vol. 45 (3): pp. 1019 – 1034, 2012. ISSN 0031-3203. doi:10.1016/j.patcog.2011.08.011. <http://www.sciencedirect.com/science/article/pii/S0031320311003487>. (p. 80, 126, 127)

(Marcus et al. 1993)

Mitchell P. MARCUS, Beatrice SANTORINI, Mary Ann MARCINKIEWICZ: *Building a Large Annotated Corpus of English: The Penn Treebank*. In: *Computational Linguistics*, vol. 19 (2): pp. 313–330, 1993. (p. 203, 204)

---

(McCallum 1999)

Andrew Kachites McCALLUM: *Multi-label text classification with a mixture model trained by EM*. In: *AAAI 99 Workshop on Text Learning*, 1999. (p. 6, 38, 172)

(McDonald et al. 2005)

Ryan T. McDONALD, Koby CRAMMER, Fernando PEREIRA: *Flexible Text Segmentation with Structured Multilabel Classification*. In: *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. The Association for Computational Linguistics, 2005. <http://acl.ldc.upenn.edu/H/H05/H05-1124.pdf>. (p. 207)

(Milgram et al. 2006)

Jonathan MILGRAM, Mohamed CHERIET, Robert SABOURIN: "One Against One" or "One Against All": Which One is Better for Handwriting Recognition with SVMs? In: *Tenth International Workshop on Frontiers in Handwriting Recognition*, ed. Guy LORETTE. Université de Rennes 1, Suvisoft, La Baule (France), 2006. <http://hal.inria.fr/inria-00103955>. (p. 70)

(Mitchell 1997)

Tom M. MITCHELL: *Machine Learning*. McGraw-Hill Science-Engineering-Math, 1997. ISBN 0070428077. (p. 4, 15, 17, 37, 97)

(Montejo Ráez et al. 2004)

Arturo MONTEJO RÁEZ, Luis Alfonso UREÑA LÓPEZ, Ralf STEINBERGER: *Adaptive Selection of Base Classifiers in One-Against-All Learning for Large Multi-labeled Collections*. In: *Advances in Natural Language Processing, 4th International Conference (EsTAL 2004)*, Alicante, Spain, October 20-22, *Proceedings*, vol. 3230 of *Lecture Notes in Computer Science*, pp. 1–12. Springer, 2004. ISBN 3-540-23498-5. [http://langtech.jrc.it/Documents/EsTAL-2004\\_Select-base-classifiers.pdf](http://langtech.jrc.it/Documents/EsTAL-2004_Select-base-classifiers.pdf). (p. 29, 47, 150, 173)

(Moreira and Mayoraz 1998)

Miguel MOREIRA, Eddy MAYORAZ: *Improved Pairwise Coupling Classification with Correcting Classifiers*. In: *Machine Learning: ECML-98, 10th European Conference on Machine Learning, Chemnitz, Germany, April 21-23, 1998, Proceedings*, vol. 1398 of *Lecture Notes in Computer Science*, eds. Claire NÉDELLEC, Céline ROUVEIROL, pp. 160–171. Springer, 1998. ISBN 3-540-64417-2. doi:10.1007/bfb0026686. (p. 72)

(Nemenyi 1963)

Peter NEMENYI: *Distribution-free multiple comparisons*. Ph.D. thesis, Princeton University, 1963. (p. 51)

(Ng et al. 1997)

Hwee T. NG, Wei B. GOH, Kok L. LOW: *Feature selection, perceptron learning, and a usability case study for text categorization*. In: *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval*, pp. 67–73. ACM

---

Press, New York, USA, 1997. <http://www.comp.nus.edu.sg/~nght/pubs/sigir97.ps.gz>. (p. 88, 93)

(Novikov 1962)

A. B. J. NOVIKOV: *On convergence proofs on Perceptrons*. In: *Proceedings of the Symposium of the Mathematical Theory of Automata*, vol. XII, pp. 615–622, 1962. (p. 87)

(Orabona et al. 2009)

Francesco ORABONA, Joseph KESHET, Barbara CAPUTO: *Bounded Kernel-Based Online Learning*. In: *Journal of Machine Learning Research*, vol. 10: pp. 2643–2666, 2009. ISSN 1532-4435. doi:10.1145/1577069.1755875. <http://jmlr.csail.mit.edu/papers/v10/orabona09a.html>. (p. 140)

(Pachet and Roy 2009)

F. PACHET, P. ROY: *Improving Multilabel Analysis of Music Titles: A Large-Scale Validation of the Correction Approach*. In: *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 17 (2): pp. 335–343, 2009. ISSN 1558-7916. doi:10.1109/tasl.2008.2008734. (p. 46, 171, 177)

(Paine 1966)

Robert T. PAINE: *Food Web Complexity and Species Diversity*. In: *The American Naturalist*, vol. 100 (910): pp. 65–75, 1966. ISSN 00030147. doi:10.2307/2459379. (p. 183)

(Park and Fürnkranz 2007)

Sang-Hyeun PARK, Johannes FÜRNKRANZ: *Efficient Pairwise Classification*. In: *Proceedings of the 18th European Conference on Machine Learning (ECML 2007, Warsaw, Poland)*, eds. J. N. KOK, J. KORONACKI, Ramon LÓPEZ DE MÁNTARAS, S. MATWIN, Dunja MLADENIĆ, A. SKOWRON, pp. 658–665. Springer-Verlag, 2007. doi:10.1007/978-3-540-74958-5\_65. <http://www.ke.tu-darmstadt.de/~juffi/publications/ecml-07-EfficientPairwise.pdf>. (p. 7, 109, 111, 113, 114, 116, 120, 121, 127)

(Park and Fürnkranz 2008)

Sang-Hyeun PARK, Johannes FÜRNKRANZ: *Multi-Label Classification with Label Constraints*. In: *Proceedings of the ECML PKDD 2008 Workshop on Preference Learning (PL-08, Antwerp, Belgium)*, eds. Eyke HÜLLERMEIER, Johannes FÜRNKRANZ, pp. 157–171, 2008. <http://www.mathematik.uni-marburg.de/~kebi/ws-ecml-08/12.pdf>. (p. 75, 175, 201, 208)

(Park and Fürnkranz 2012)

Sang-Hyeun PARK, Johannes FÜRNKRANZ: *Efficient prediction algorithms for binary decomposition techniques*. In: *Data Mining and Knowledge Discovery*, vol. 24 (1): pp. 40–77, 2012. ISSN 1384-5810. doi:10.1007/s10618-011-0219-9. <http://www.ke.tu-darmstadt.de/publications/papers/dami12.pdf>. (p. 115)

(Peters 2009)

Isabella PETERS: *Folksonomies: Indexing and Retrieval in the Web 2.0*. De Gruyter Saur, 2009. ISBN 978-3598251795. (p. 43)

---

(Petrovskiy 2006)

Mikhail PETROVSKIY: *Paired Comparisons Method for Solving Multi-Label Learning Problem*. In: *Proceedings of the Sixth International Conference on Hybrid Intelligent Systems, HIS '06*. IEEE Computer Society, Washington, DC, USA, 2006. ISBN 0-7695-2662-4. doi:10.1109/his.2006.54. (p. 72)

(Petrovskiy and Glazkova 2006)

Mikhail PETROVSKIY, Valentina GLAZKOVA: *Linear Methods for Reduction from Ranking to Multilabel Classification*. In: *AI 2006: Advances in Artificial Intelligence*, vol. 4304 of *Lecture Notes in Computer Science*, eds. Abdul SATTAR, Byeong-ho KANG, pp. 1152–1156. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-49787-5. doi:10.1007/11941439\_139. (p. 74, 75)

(Platt 2000)

John C. PLATT: *Probabilistic outputs for support vector machines and comparison to regularize likelihood methods*. In: *Advances in Large Margin Classifiers*, eds. Alexander J. SMOLA, Peter L. BARTLETT, Bernhard SCHÖLKOPF, D. SCHUURMANS, pp. 61–74, 2000. <http://research.microsoft.com/~jplatt/SVMprob.ps.gz>. (p. 77)

(Pouliquen et al. 2003)

Bruno POULIQUEN, Ralf STEINBERGER, Camelia IGNAT: *Automatic annotation of multilingual text collections with a conceptual thesaurus*. In: *Proceedings of the Workshop 'Ontologies and Information Extraction' at the Summer School 'The Semantic Web and Language Technology - Its Potential and Practicalities' (EUROLAN'2003)*, 28 July - 8 August 2003, pp. 9–28. Bucharest, Romania, 2003. [http://langtech.jrc.it/Documents/EuroLan-03\\_Pouliquen-Steinberger-et-al.pdf](http://langtech.jrc.it/Documents/EuroLan-03_Pouliquen-Steinberger-et-al.pdf). (p. 133, 150, 173)

(Pratt 1959)

John W. PRATT: *Remarks on Zeros and Ties in the Wilcoxon Signed Rank Procedures*. In: *Journal of the American Statistical Association*, vol. 54 (287): pp. 655–667, 1959. ISSN 01621459. <http://www.jstor.org/stable/2282543>. (p. 50)

(Price et al. 1994)

David PRICE, Stefan KNERR, Léon PERSONNAZ, Gérard DREYFUS: *Pairwise Neural Network Classifiers with Probabilistic Outputs*. In: *Advances in Neural Information Processing Systems 7 (NIPS-94)*, vol. 7, eds. Gerald TESAURO, D. TOURETZKY, Todd K. LEEN, pp. 1109–1116. MIT Press, 1994. [http://www.neurones.espci.fr/Articles\\_PS/knerr.pdf](http://www.neurones.espci.fr/Articles_PS/knerr.pdf). (p. 77)

(Putter 1955)

Joseph PUTTER: *The Treatment of Ties in Some Nonparametric Tests*. In: *The Annals of Mathematical Statistics*, vol. 26 (3): pp. 368–386, 1955. doi:10.1214/aoms/1177728485. <http://projecteuclid.org/euclid.aoms/1177728485>. (p. 49)

(Read 2012)

Jesse READ: *MEKA: A Multi-label Extension to the WEKA Machine Learning Framework*,



---

Dataset Repository. Website, 2012. <http://meka.sourceforge.net>. Last accessed at 2012-01-12. (p. 46, 47)

(Read et al. 2008)

Jesse READ, Bernhard PFAHRINGER, Geoff HOLMES: *Multi-label Classification Using Ensembles of Pruned Sets*. In: *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pp. 995–1000. IEEE Computer Society, Washington, DC, USA, 2008. ISBN 978-0-7695-3502-9. doi:10.1109/icdm.2008.74. <http://www.tsc.uc3m.es/~jesse/papers/icdm08-eps-long.pdf>. (p. 39)

(Read et al. 2009)

Jesse READ, Bernhard PFAHRINGER, Geoff HOLMES, Eibe FRANK: *Classifier Chains for Multi-label Classification*. In: *Machine Learning and Knowledge Discovery in Databases*, vol. 5782 of *Lecture Notes in Computer Science*, eds. Wray BUNTINE, Marko GROBELNIK, Dunja MLADENIĆ, John SHAWE-TAYLOR, pp. 254–269. Springer Berlin / Heidelberg, 2009. doi:10.1007/978-3-642-04174-7\_17. <http://www.cs.waikato.ac.nz/~eibe/pubs/chains.pdf>. (p. 39, 79, 80, 172, 191, 200)

(Read et al. 2010)

Jesse READ, Albert BIFET, Geoffrey HOLMES, Bernhard PFAHRINGER: *Efficient multi-label classification for evolving data streams*. Tech. rep., University of Waikato, Department of Computer Science, 2010. (p. 38, 103)

(Read et al. 2011)

Jesse READ, Bernhard PFAHRINGER, Geoff HOLMES, Eibe FRANK: *Classifier chains for multi-label classification*. In: *Machine Learning*, vol. 85 (3): pp. 333–359, 2011. ISSN 0885-6125. doi:10.1007/s10994-011-5256-5. (p. 39, 40, 79, 186)

(Rifkin and Klautau 2004)

Ryan RIFKIN, Aldebaro KLAUTAU: *In Defense of One-Vs-All Classification*. In: *Journal of Machine Learning Research*, vol. 5: pp. 101–141, 2004. <http://jmlr.csail.mit.edu/papers/v5/rifkin04a.html>. (p. 37, 70, 79, 80)

(Rosenblatt 1958)

Frank ROSENBLATT: *The perceptron: a probabilistic model for information storage and organization in the brain*. In: *Psychological Review*, vol. 65 (6): pp. 386–408, 1958. [http://www.ling.upenn.edu/courses/Fall\\_2007/cogs501/Rosenblatt1958.pdf](http://www.ling.upenn.edu/courses/Fall_2007/cogs501/Rosenblatt1958.pdf). (p. 84)

(Rosenblatt 1988)

Frank ROSENBLATT: *The perceptron: A probabilistic model for information storage and organization in the brain*. In: *Neurocomputing: foundations of research*, eds. James ANDERSON, Edward ROSENFELD, pp. 89–114. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-01097-6. Reprint of 1958 article in *Psychological Review*. (p. 84)

(Rubin et al. 2011)

Timothy N. RUBIN, America CHAMBERS, Padhraic SMYTH, Mark STEYVERS: *Statistical Topic*



- 
- Models for Multi-Label Document Classification*. Tech. Rep. arXiv:1107.2462, Dept. of Cognitive Sciences at UC Irvine, Memory and Decision-Making Laboratory, 2011. <http://arxiv.org/abs/1107.2462>. To be published in: The Machine Learning Journal, special issue on Learning from Multi-Label Data. (p. 25, 32, 33, 39, 134, 136, 150, 151)
- (Rueda et al. 2010)
- Luis RUEDA, B. John OOMMEN, Claudio HENRÍQUEZ: *Multi-class pairwise linear dimensionality reduction using heteroscedastic schemes*. In: *Pattern Recognition*, vol. 43 (7): pp. 2456–2465, 2010. ISSN 0031-3203. doi:10.1016/j.patcog.2010.01.018. [http://brage.bibsys.no/hia/bitstream/URN:NBN:no-bibsys\\_brage\\_16224/1/Oommen\\_2010\\_Multi.pdf](http://brage.bibsys.no/hia/bitstream/URN:NBN:no-bibsys_brage_16224/1/Oommen_2010_Multi.pdf). (p. 37, 41)
- (Salton and Buckley 1988)
- Gerard SALTON, Chris BUCKLEY: *Term-Weighting Approaches in Automatic Text Retrieval*. In: *Information Processing and Management*, vol. 24 (5): pp. 513–523, 1988. doi:10.1016/0306-4573(88)90021-0. (p. 45, 96, 136)
- (Schapire and Singer 1999)
- Robert E. SCHAPIRE, Yoram SINGER: *Improved Boosting Algorithms Using Confidence-rated Predictions*. In: *Machine Learning*, vol. 37 (3): pp. 297–336, 1999. (p. 39)
- (Schapire and Singer 2000)
- Robert E. SCHAPIRE, Yoram SINGER: *BoosTexter: A Boosting-based System for Text Categorization*. In: *Machine Learning*, vol. 39 (2/3): pp. 135–168, 2000. doi:10.1023/a:1007649029923. <http://www.cs.princeton.edu/~schapire/papers/SchapireSi98b.ps>. (p. 39)
- (Sebastiani 2002)
- Fabrizio SEBASTIANI: *Machine learning in automated text categorization*. In: *ACM Computing Surveys*, vol. 34 (1): pp. 1–47, 2002. <http://nmis.isti.cnr.it/sebastiani/Publications/ACMCS02.pdf>. (p. 4, 18, 29, 42, 86, 88, 90, 96, 136)
- (Shalev-Shwartz and Singer 2005)
- Shai SHALEV-SHWARTZ, Yoram SINGER: *A New Perspective on an Old Perceptron Algorithm*. In: *Learning Theory, 18th Annual Conference on Learning Theory (COLT 2005)*, pp. 264–278. Springer, 2005. ISBN 3-540-26556-2. doi:10.1007/11503415\_18. <http://www.cs.huji.ac.il/~shais/papers/ShalevSi05.pdf>. (p. 89)
- (Shapiro and Haralick 1985)
- Linda G. SHAPIRO, Robert M. HARALICK: *A Metric for Comparing Relational Descriptions*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7: pp. 90–94, 1985. ISSN 0162-8828. (p. 189)
- (Shi et al. 2011)
- Chuan SHI, Xiangnan KONG, Philip S. YU, Bai WANG: *Multi-label ensemble learning*. In:

---

*Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part III, ECML PKDD'11*, pp. 223–239. Springer-Verlag, Berlin, Heidelberg, 2011. ISBN 978-3-642-23807-9. (p. 40)

(Sima 2008)

Jan Frederik SIMA: *Paarweise Hierarchische Klassifikation*. Master's thesis, TU Darmstadt, Knowledge Engineering Group, 2008. [http://www.ke.informatik.tu-darmstadt.de/lehre/arbeiten/diplom/2008/Sima\\_Jan-Frederik.pdf](http://www.ke.informatik.tu-darmstadt.de/lehre/arbeiten/diplom/2008/Sima_Jan-Frederik.pdf). Diplom. (p. 23)

(Snoek et al. 2006)

Cees G. M. SNOEK, Marcel WORRING, Jan C. VAN GEMERT, Jan-Mark GEUSEBROEK, Arnold W. M. SMEULDERS: *The Challenge Problem for Automated Detection of 101 Semantic Concepts in Multimedia*. In: *Proceedings of ACM Multimedia*, pp. 421–430. Santa Barbara, CA, 2006. (p. 46)

(Sokolova and Guy 2009)

Marina SOKOLOVA, Lapal GUY: *A systematic analysis of performance measures for classification tasks*. In: *Journal Information Processing and Management*, vol. 45: pp. 427–437, 2009. ISSN 0306-4573. doi:10.1016/j.ipm.2009.03.002. <http://www-rali.iro.umontreal.ca/Publications/files/SokolovaLapalme-JIPM09.pdf>. (p. 35)

(Spyromitros Xioufis et al. 2011)

Eleftherios SPYROMITROS XIOUFIS, Myra SPILIOPOULOU, Grigorios TSOUMAKAS, Ioannis P. VLAHAVAS: *Dealing with Concept Drift and Class Imbalance in Multi-label Stream Classification*. In: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pp. 1583–1588, 2011. ISBN 978-1-57735-516-8. <http://ijcai.org/papers11/Papers/IJCAI11-266.pdf>. (p. 103)

(Srivastava and Zane-Ulman 2005)

Ashok N. SRIVASTAVA, Brett ZANE-ULMAN: *Discovering recurring anomalies in text reports regarding complex space systems*. In: *IEEE Aerospace Conference*, pp. 3853–3862, 2005. ISBN 0-7803-8870-4. doi:0.1109/AERO.2005.1559692. <https://c3.nasa.gov/dashlink/static/media/publication/IEEE-Aero-2005.pdf>. (p. 46)

(Steinberger et al. 2002)

Ralf STEINBERGER, Bruno POULIQUEN, Johan HAGMAN: *Cross-Lingual Document Similarity Calculation Using the Multilingual Thesaurus EUROVOC*. In: *CICLing '02: Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, pp. 415–424. Springer, London, UK, 2002. ISBN 3-540-43219-1. doi: 10.1007/3-540-45715-1\_44. [http://langtech.jrc.it/documents/cicling-02\\_steinberger.pdf](http://langtech.jrc.it/documents/cicling-02_steinberger.pdf). (p. 133, 150)

(Streich and Buhmann 2008)

Andreas P. STREICH, Joachim M. BUHMANN: *Classification of Multi-labeled Data: A Generative Approach*. In: *Proceedings ECML PKDD '08 Proceedings of the European con-*

- 
- ference on Machine Learning and Knowledge Discovery in Databases - Part II*, vol. 5212 of *Lecture Notes in Computer Science*, eds. Walter DAELEMANS, Bart GOETHALS, Katharina MORIK, pp. 390–405. Springer, 2008. ISBN 978-3-540-87480-5. doi: 10.1007/978-3-540-87481-2\_26. [http://www.inf.ethz.ch/personal/astreich/papers/multilabel\\_generative\\_ECML2008.pdf](http://www.inf.ethz.ch/personal/astreich/papers/multilabel_generative_ECML2008.pdf). (p. 39)
- (Sulzmann and Fürnkranz 2008)  
Jan-Nikolas SULZMANN, Johannes FÜRNKRANZ: *A Comparison of Techniques for Selecting and Combining Class Association Rules*. In: *From Local Patterns to Global Models: Proceedings of the ECML/PKDD-08 Workshop (LeGo-08)*, ed. Arno J. KNOBBE, pp. 154–168. Antwerp, Belgium, 2008. <http://www.ke.tu-darmstadt.de/events/LeGo-08/11.pdf>. (p. 185)
- (Sulzmann et al. 2007)  
Jan-Nikolas SULZMANN, Johannes FÜRNKRANZ, Eyke HÜLLERMEIER: *On Pairwise Naive Bayes Classifiers*. In: *Proceedings of 18th European Conference on Machine Learning (ECML-07)*, eds. J. N. KOK, J. KORONACKI, Ramon LÓPEZ DE MÁNTARAS, S. MATWIN, Dunja MLADENIĆ, A. SKOWRON, pp. 371–381. Springer-Verlag, Warsaw, Poland, 2007. <http://www.ke.tu-darmstadt.de/~juffi/publications/ecml-07-NaiveBayes.pdf>. (p. 97)
- (Sun et al. 2008)  
Liang SUN, Shuiwang JI, Jieping YE: *Hypergraph spectral learning for multi-label classification*. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '08*, pp. 668–676. ACM, New York, NY, USA, 2008. (p. 41)
- (Tai and Lin 2010)  
Farbound TAI, Hsuan-tien LIN: *Multi-label Classification with Principle Label Space Transformation*. In: Zhang et al. (2010a), pp. 45–52. <http://www.csie.ntu.edu.tw/~htlin/paper/doc/wsml10plst.pdf>. An extended version under <http://www.csie.ntu.edu.tw/~htlin/paper/doc/plst.pdf> is to appear in *Neural Computation*. (p. 41, 150)
- (Tang et al. 2009)  
Lei TANG, Suju RAJAN, Vijay K. NARAYANAN: *Large Scale Multi-Label Classification via MetaLabeler*. In: *18th International World Wide Web Conference*, pp. 211–211, 2009. ISBN 978-1-60558-487-4. doi:10.1145/1526709.1526738. <http://www2009.eprints.org/22/>. (p. 74, 173)
- (Thabtah et al. 2006)  
Fadi A. THABTAH, Peter I. COWLING, Yonghong PENG: *Multiple labels associative classification*. In: *Knowledge and Information Systems*, vol. 9 (1): pp. 109–129, 2006. doi: 10.1007/s10115-005-0213-x. <http://selab.iecs.fcu.edu.tw/course/95/dp2/dpclass/16.pdf>. (p. 38)
- (Thurstone 1927)  
Louis Leon THURSTONE: *A Law of Comparative Judgement*. In: *Psychological Re-*

---

view, vol. 34: pp. 278–286, 1927. [http://www.brocku.ca/MeadProject/Thurstone/Thurstone\\_1929a.html](http://www.brocku.ca/MeadProject/Thurstone/Thurstone_1929a.html). (p. 2)

(Trohidis et al. 2008)

Konstantinos TROHIDIS, Grigorios TSOUMAKAS, George KALLIRIS, Ioannis P. VLAHAVAS: *Multilabel Classification of Music into Emotions*. In: *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008)*, pp. 325–330, 2008. <http://lps.csd.auth.gr/publications/tsoumakas-ismir08.pdf>. (p. 46)

(Tsampouka and Shawe-Taylor 2007)

Petroula TSAMPOUKA, John SHAWE-TAYLOR: *Approximate maximum margin algorithms with rules controlled by the number of mistakes*. In: *Machine Learning, Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML 2007)*, vol. 227 of *ACM International Conference Proceeding Series*, ed. Zoubin GHAHRAMANI, pp. 903–910. ACM, 2007. ISBN 978-1-59593-793-3. doi:10.1145/1273496.1273610. (p. 89)

(Tsochantaridis and Hofmann 2002)

Ioannis TSOCHANTARIDIS, Thomas HOFMANN: *Support Vector Machines for Polycategorical Classification*. In: *Proceedings of the 13th European Conference on Machine Learning*, pp. 456–467. Springer-Verlag, London, UK, 2002. ISBN 3-540-44036-4. (p. 41)

(Tsochantaridis et al. 2005)

Ioannis TSOCHANTARIDIS, Thorsten JOACHIMS, Thomas HOFMANN, Yasemin ALTUN: *Large Margin Methods for Structured and Interdependent Output Variables*. In: *Journal of Machine Learning Research*, vol. 6: pp. 1453–1484, 2005. ISSN 1532-4435. (p. 40, 172)

(Tsoumakas 2012)

Grigorios TSOUMAKAS: *Mulan: A Java Library for Multi-Label Learning, Dataset Repository*. Website, 2012. <http://mulan.sourceforge.net/datasets.html>. Last accessed at 2012-01-12. (p. 46, 47, 109)

(Tsoumakas and Katakis 2007)

Grigorios TSOUMAKAS, Ioannis KATAKIS: *Multi-Label Classification: An Overview*. In: *International Journal of Data Warehousing and Mining*, vol. 3 (3): pp. 1–17, 2007. <http://lps.csd.auth.gr/publications/tsoumakas-ijdwm.pdf>. (p. 29, 36, 37, 53, 54, 57)

(Tsoumakas and Vlahavas 2007)

Grigorios TSOUMAKAS, Ioannis P. VLAHAVAS: *Random K-Labelsets: An Ensemble Method for Multilabel Classification*. In: *Machine Learning: ECML 2007, 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007, Proceedings*, vol. 4701 of *Lecture Notes in Computer Science*, eds. Joost N. KOK, Jacek KORONACKI, Ramon LÓPEZ DE MÁNTARAS, Stan MATWIN, Dunja MLADENIĆ, Andrzej SKOWRON, pp. 406–417. Springer, 2007. ISBN 978-3-540-74957-8. doi:10.1007/978-3-540-74958-5\_38. (p. 30)

(Tsoumakas et al. 2008)

Grigorios TSOUMAKAS, I. KATAKIS, Ioannis P. VLAHAVAS: *Effective and Efficient Multilabel*

- 
- Classification in Domains with Large Number of Labels*. In: *Proceedings ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, 2008. (p. 8, 43, 46, 150, 153, 154, 155, 156, 160, 161, 167, 169, 177, 210)
- (Tsoumakas et al. 2009)  
 Grigorios TSOUMAKAS, Anastasios DIMOU, Eleftherios SPYROMITROS XIOUFIS, Vasileios MEZARIS, Ioannis KOMPATSIARIS, Ioannis P. VLAHAVAS: *Correlation Based Pruning of Stacked Binary Relevance Models for Multi-Label Learning*. In: Tsoumakas et al. (2009c), pp. 101–116. <http://lps.csd.auth.gr/workshops/mld09/mld09.pdf>. (p. 200)
- (Tsoumakas et al. 2009)  
 Grigorios TSOUMAKAS, Eneldo LOZA MENCÍA, Ioannis KATAKIS, Sang-Hyeun PARK, Johannes FÜRNKRANZ: *On the Combination of Two Decompositive Multi-Label Classification Methods*. In: *Proceedings of the ECML PKDD 2009 Workshop on Preference Learning (PL-09, Bled, Slovenia)*, eds. Eyke HÜLLERMEIER, Johannes FÜRNKRANZ, pp. 114–129, 2009b. <http://www.ke.tu-darmstadt.de/events/PL-09/09-Tsoumakas.pdf>. (p. 8, 153)
- (Tsoumakas et al. 2009)  
 Grigorios TSOUMAKAS, Min-Ling ZHANG, Zhi-Hua ZHOU (eds.): *Proceedings of the 1st International Workshop on Learning from Multi-Label Data (MLD'09)*, 2009c. <http://lps.csd.auth.gr/workshops/mld09/mld09.pdf>. (p. 17, 236)
- (Tsoumakas et al. 2010)  
 Grigorios TSOUMAKAS, Ioannis KATAKIS, Ioannis P. VLAHAVAS: *Mining Multi-label Data*. In: *Data Mining and Knowledge Discovery Handbook*, eds. Oded MAIMON, Lior ROKACH, pp. 667–685. Springer, 2010. ISBN 978-0-387-09823-4. doi:10.1007/978-0-387-09823-4\_34. <http://lps.csd.auth.gr/publications/tsoumakas09-dmkydh.pdf>. (p. 33, 37, 53)
- (Tsoumakas et al. 2011)  
 Grigorios TSOUMAKAS, Ioannis KATAKIS, Ioannis P. VLAHAVAS: *Random k-Labelsets for Multilabel Classification*. In: *IEEE Transactions on Knowledge and Data Engineering*, vol. 23 (7): pp. 1079–1089, 2011a. doi:10.1109/tkde.2010.164. (p. 39)
- (Tsoumakas et al. 2011)  
 Grigorios TSOUMAKAS, Eleftherios SPYROMITROS XIOUFIS, Jozef VILCEK, Ioannis P. VLAHAVAS: *Mulan: A Java Library for Multi-Label Learning*. In: *Journal of Machine Learning Research*, vol. 12: pp. 2411–2414, 2011b. <http://jmlr.csail.mit.edu/papers/volume12/tsoumakas11a/tsoumakas11a.pdf>. Software available at <http://mulan.sourceforge.net/>. (p. 47, 127, 160, 191, 201)
- (Ueda and Saito 2002)  
 Naonori UEDA, Kazumi SAITO: *Parametric Mixture Models for Multi-Labeled Text*. In: *Advances in Neural Information Processing Systems (NIPS) 15*, eds. Suzanna BECKER, Sebastian THRUN, Klaus OBERMAYER, NIPS, pp. 721–728, 2002. ISBN 0-262-02550-7. <http://books.nips.cc/papers/files/nips15/AA30.pdf>. (p. 38, 47)

---

(Veloso et al. 2007)

Adriano VELOSO, Wagner MEIRA, Jr., Marcos André GONÇALVES, Mohammed ZAKI: *Multi-label Lazy Associative Classification*. In: *Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, vol. 4702 of *Lecture Notes in Computer Science*, eds. Joost N. KOK, Jacek KORONACKI, Ramon LÓPEZ DE MÁNTARAS, Stan MATWIN, Dunja MLADENIĆ, Andrzej SKOWRON, pp. 605–612. Springer, 2007. ISBN 978-3-540-74975-2. doi:10.1007/978-3-540-74976-9\_64. <http://www.dcc.ufmg.br/~adrianov/papers/PKDD07/Veloso-pkdd07.pdf>. (p. 38, 107, 199)

(Vens et al. 2008)

Celine VENS, Jan STRUYF, Leander SCHIETGAT, Sašo DŽEROSKI, Hendrik BLOCCKEEL: *Decision Trees for Hierarchical Multi-Label Classification*. In: *Machine Learning*, vol. 73 (2): pp. 185–214, 2008. <https://lirias.kuleuven.be/bitstream/123456789/186698/4/hmc.pdf>. (p. 23, 168)

(Verma and Pearl 1990)

Thomas VERMA, Judea PEARL: *Equivalence and synthesis of causal models*. In: *UAI '90 Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 255–270. Elsevier Science Inc., 1990. ISBN 0-444-89264-8. (p. 189)

(Villalba and Cunningham 2009)

Santiago D. VILLALBA, P. CUNNINGHAM: *Using Unsupervised Classifiers for Multilabel Classification in Open-Class-Set Scenarios*. In: *Proceedings of the ECML/PKDD-09 Workshop on Learning from Multi-Label Data (MLD-09)*, eds. Grigorios TSOUAKAS, Min-Ling ZHANG, Zhi-Hua ZHOU, pp. 146–160. Bled, Slovenia, 2009. <http://lps.csd.auth.gr/workshops/mld09/mld09.pdf>. (p. 173)

(Wang et al. 2011)

H. WANG, Xiaotong SHEN, Wei PAN: *Large Margin Hierarchical Classification with Mutually Exclusive Class Membership*. In: *Journal of Machine Learning Research*, vol. 12: pp. 2721–2748, 2011. <http://jmlr.csail.mit.edu/papers/v12/wang11c.html>. (p. 22)

(Weiss et al. 2008)

Yair WEISS, Antonio B. TORRALBA, Robert FERGUS: *Spectral Hashing*. In: *Advances in Neural Information Processing Systems (NIPS)*, eds. Daphne KOLLER, Dale SCHUURMANS, Yoshua BENGIO, Léon BOTTOU, pp. 1753–1760. MIT Press, 2008. <http://people.csail.mit.edu/torralba/publications/spectralhashing.pdf>. (p. 41)

(Wilcoxon 1950)

Frank WILCOXON: *Some rapid approximate statistical procedures*. In: *Annals of the New York Academy of Sciences*, vol. 52: pp. 808–814, 1950. doi:10.1111/j.1749-6632.1950.tb53974.x. (p. 49)

(Witten and Frank 2005)

Ian H. WITTEN, Eibe FRANK: *Data mining: Practical machine learning tools and tech-*



- 
- niques*. 2 ed. Morgan Kaufmann Publishers, San Francisco, 2005. ISBN 0-12-088407-0. <http://www.cs.waikato.ac.nz/~ml/weka/>. (p. 2, 37, 38, 39, 56, 73, 149, 156, 160, 177, 192, 200)
- (Wu et al. 2004)  
 Ting-Fan WU, Chih-Jen LIN, Ruby C. WENG: *Probability Estimates for Multi-class Classification by Pairwise Coupling*. In: *Journal of Machine Learning Research*, vol. 5: pp. 975–1005, 2004. <http://www.jmlr.org/papers/volume5/wu04a/wu04a.pdf>. (p. 77, 78)
- (Xue et al. 2007)  
 Ya XUE, Xuejun LIAO, Lawrence CARIN, Balaji KRISHNAPURAM: *Multi-task learning for classification with dirichlet process priors*. In: *Journal of Machine Learning Research*, vol. 8: pp. 35–63, 2007. <http://www.jmlr.org/papers/v8/xue07a.html>. (p. 172)
- (Yang and Liu 1999)  
 Y. YANG, X. LIU: *A re-examination of text categorization methods*. In: *22nd Annual International SIGIR*, pp. 42–49. Berkley, 1999. (p. 29)
- (Yang 1999)  
 Yiming YANG: *An Evaluation of Statistical Approaches to Text Categorization*. In: *Information Retrieval*, vol. 1: pp. 69–90, 1999. (p. 73)
- (Yang and Pedersen 1997)  
 Yiming YANG, Jan O. PEDERSEN: *A Comparative Study on Feature Selection in Text Categorization*. In: *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 412–420. Morgan Kaufmann Publishers Inc., 1997. ISBN 1-55860-486-3. <http://www.cs.cmu.edu/~yiming/publications.html>. (p. 136)
- (Zhang 2009)  
 Min-Ling ZHANG: *ML-RBF: RBF Neural Networks for Multi-Label Learning*. In: *Neural Processing Letters*, vol. 29: pp. 61–74, 2009. ISSN 1370-4621. doi:10.1007/s11063-009-9095-3. <http://cse.seu.edu.cn/people/zhangml/files/NPL09.pdf>. (p. 39)
- (Zhang 2011)  
 Min-Ling ZHANG: *LIFT: Multi-Label Learning with Label-Specific Features*. In: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 1609–1614, 2011. <http://ijcai.org/papers11/Papers/IJCAI11-270.pdf>. (p. 40, 200)
- (Zhang and Zhang 2010)  
 Min-Ling ZHANG, Kun ZHANG: *Multi-label learning by exploiting label dependency*. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '10*, pp. 999–1008. ACM, New York, NY, USA, 2010. ISBN 978-1-4503-0055-1. doi:10.1145/1835804.1835930. (p. 24, 200)



---

(Zhang and Zhou 2006)

Min-Ling ZHANG, Zhi-Hua ZHOU: *Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization*. In: *IEEE Transactions on Knowledge and Data Engineering*, vol. 18: pp. 1338–1351, 2006. ISSN 1041-4347. doi:10.1109/tkde.2006.162. <http://cs.nju.edu.cn/zhoush/zhoush.files/publication/tkde06a.pdf>. (p. 38, 107, 172)

(Zhang and Zhou 2007)

Min-Ling ZHANG, Zhi-Hua ZHOU: *ML-KNN: A lazy learning approach to multi-label learning*. In: *Pattern Recognition*, vol. 40: pp. 2038–2048, 2007. ISSN 0031-3203. doi:10.1016/j.patcog.2006.12.019. (p. 38)

(Zhang et al. 2010)

Min-Ling ZHANG, Grigorios TSOU MAKAS, Zhi-Hua ZHOU (eds.): *Working Notes of the 2nd International Workshop on Learning from Multi-Label Data at ICML/COLT 2010*, 2010a. <http://cse.seu.edu.cn/conf/MLD10/files/MLD'10.pdf>. (p. 17, 226, 234)

(Zhang et al. 2010)

Xiatian ZHANG, Quan YUAN, Shiwan ZHAO, Wei FAN, Wentao ZHENG, Zhong WANG: *Multi-label Classification without the Multi-label cost*. In: *Proceedings of the Tenth SIAM International Conference on Data Mining*, 2010b. [https://www.siam.org/proceedings/datamining/2010/dm10\\_068\\_zhangx.pdf](https://www.siam.org/proceedings/datamining/2010/dm10_068_zhangx.pdf). (p. 38, 103, 104)

(Zhu et al. 2005)

Shenghuo ZHU, Xiang JI, Wei XU, Yihong GONG: *Multi-labelled classification using maximum entropy method*. In: *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 274–281. ACM, New York, NY, USA, 2005. ISBN 1-59593-034-5. doi:10.1145/1076034.1076082. (p. 30, 172)

(Zimek et al. 2010)

Arthur ZIMEK, Fabian BUCHWALD, Eibe FRANK, Stefan KRAMER: *A Study of Hierarchical and Flat Classification of Proteins*. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 7: pp. 563–571, 2010. <http://waikato.researchgateway.ac.nz/handle/10289/2677>. (p. 23, 168)



---

# Own Publications

## Journal Papers

Johannes FÜRNKRANZ, Eyke HÜLLERMEIER, Eneldo LOZA MENCÍA, Klaus BRINKER: *Multilabel Classification via Calibrated Label Ranking*. In: *Machine Learning*, vol. 73 (2): pp. 133–153, 2008. doi:10.1007/s10994-008-5064-8. <http://www.ke.tu-darmstadt.de/publications/papers/MLJ08.pdf>. (p. 7, 33, 34, 63, 64, 65, 66, 67, 70, 84, 105)

Eneldo LOZA MENCÍA, Sang-Hyeun PARK, Johannes FÜRNKRANZ: *Efficient Voting Prediction for Pairwise Multilabel Classification*. In: *Neurocomputing*, vol. 73 (7-9): pp. 1164 – 1176, 2010. ISSN 0925-2312. doi:10.1016/j.neucom.2009.11.024. <http://www.ke.tu-darmstadt.de/publications/papers/neucom10.pdf>. (p. 7, 107, 123)

## Book Chapters

Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ: *Efficient Multilabel Classification Algorithms for Large-Scale Problems in the Legal Domain*. In: *Semantic Processing of Legal Texts – Where the Language of Law Meets the Law of Language*, vol. 6036 of *Lecture Notes in Artificial Intelligence*, eds. Enrico FRANCESCONI, Simonetta MONTEMAGNI, Wim PETERS, Daniela TISCORNIA, 1 ed., pp. 192–215. Springer-Verlag, 2010. ISBN 978-3-642-12836-3. doi:10.1007/978-3-642-12837-0\_11. <http://www.ke.tu-darmstadt.de/publications/papers/loza10eurlex.pdf>. Accompanying EUR-Lex dataset available at <http://www.ke.tu-darmstadt.de/resources/eurlex>. (p. 8, 47, 107)

## Conference Papers

Wouter DUIVESTIJN, Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ, Arno J. KNOBBE: *Multi-label LeGo – Enhancing Multi-label Classifiers with Local Patterns*. In: *Advances in Intelligent Data Analysis XI – Proceedings of the 11th International Symposium on Data Analysis (IDA-11)*, vol. 7619 of *Lecture Notes in Computer Science*, eds. Jaakko HOLLMÉN, Frank KLAWONN, Allan TUCKER, pp. 114–125. Springer, 2012. ISBN 978-3-642-34155-7. doi:10.1007/978-3-642-34156-4\_12. <http://www.ke.tu-darmstadt.de/publications/papers/IDA-12.pdf>. (p. 8, 199)

---

Page number(s) in parenthesis at the end indicate the place(s) where each entry was referenced.

---

Eneldo LOZA MENCÍA: *Segmentation of legal documents*. In: *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pp. 88–97. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-597-0. doi:10.1145/1568234.1568245. <http://www.ke.tu-darmstadt.de/publications/papers/loza09segmentation.pdf>. (p. 204)

Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ: *Efficient Pairwise Multilabel Classification for Large-Scale Problems in the Legal Domain*. In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD-2008), Part II*, vol. 5212 of *Lecture Notes in Computer Science*, eds. Walter DAELEMANS, Bart GOETHALS, Katharina MORIK, pp. 50–65. Springer, Antwerp, Belgium, 2008a. ISBN 978-3-540-87480-5. doi:10.1007/978-3-540-87481-2\_4. <http://www.ke.tu-darmstadt.de/publications/papers/ECML08.pdf>. Accompanying EUR-Lex dataset available at <http://www.ke.tu-darmstadt.de/resources/eurllex>. (p. 8, 132, 143, 151)

Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ: *Pairwise Learning of Multilabel Classifications with Perceptrons*. In: *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IJCNN-08)*, pp. 2900–2907. IEEE, Hong Kong, 2008b. ISBN 978-1-4244-1821-3. doi:10.1109/IJCNN.2008.4634206. (p. 7, 84, 105)

Eneldo LOZA MENCÍA, Sang-Hyeun PARK, Johannes FÜRNKRANZ: *Efficient Voting Prediction for Pairwise Multilabel Classification*. In: *Proceedings of the 17th European Symposium on Artificial Neural Networks (ESANN 2009, Bruges, Belgium)*, pp. 117–122. d-side publications, 2009. ISBN 2-930307-09-9. <http://www.dice.ucl.ac.be/Proceedings/esann/esannpdf/es2009-112.pdf>. (p. 7)

## Workshop Papers

Eneldo LOZA MENCÍA: *An Evaluation of Multilabel Classification for the Automatic Annotation of Texts*. In: *Proceedings of the LWA 2010: Lernen, Wissen, Adaptivität, Workshop on Knowledge Discovery, Data Mining and Machine Learning (KDML 2010)*, eds. Martin ATZMÜLLER, Dominik BENZ, Andreas HOTH, Gerd STUMME, pp. 121–123, 2010a. <http://www.kde.cs.uni-kassel.de/conf/lwa10/papers/kdml11.pdf>. (p. 9)

Eneldo LOZA MENCÍA: *Multilabel Classification in Parallel Tasks*. In: *Working Notes of the 2nd International Workshop on Learning from Multi-Label Data at ICML/COLT 2010*, eds. Min-Ling ZHANG, Grigorios TSOU MAKAS, Zhi-Hua ZHOU, pp. 29–36, 2010b. <http://www.ke.tu-darmstadt.de/publications/papers/loza10mlpt.pdf>. (p. 8)

Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ: *An Evaluation of Efficient Multilabel Classification Algorithms for Large-Scale Problems in the Legal Domain*. In: *Proceedings of the LWA 2007: Lernen - Wissen - Adaption*, ed. Alexander HINNEBURG, pp. 126–132. Halle, Germany, 2007. ISBN 978-3-86010-907-6. <http://www.ke.tu-darmstadt.de/publications/papers/LWA07.pdf>. (p. 8, 45, 46, 148)

---

Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ: *An Evaluation of Efficient Multilabel Classification Algorithms for Large-Scale Problems in the Legal Domain*. In: *Proceedings of the LREC 2008 Workshop on Semantic Processing of Legal Texts*, eds. Simonetta MONTMAGNI, Daniela TISCORNIA, Enrico FRANCESCONI, Wim PETERS, pp. 23–32. Marrakech, Morocco, 2008. <http://www.ke.tu-darmstadt.de/publications/papers/LREC2008.pdf>. (p. 132, 143)

Eneldo LOZA MENCÍA, Sang-Hyeun PARK, Johannes FÜRNKRANZ: *Efficient Voting Prediction for Pairwise Multilabel Classification*. In: *Proceedings of the LWA 2009: Lernen - Wissen - Adaption, Workshop Knowledge Discovery, Data Mining and Machine Learning (KDML-09)*, eds. Dominik BENZ, Frederik JANSSEN, Knowledge Engineering Group Technical Report TUD-KE-2009-04, Telekooperation Group Technical Report TUD-CS-2009-0157, pp. 72–75. Darmstadt, Germany, 2009. <http://www.ke.tu-darmstadt.de/publications/papers/KDML09.pdf>. Resubmission. (p. )

Grigorios TSOUMAKAS, Eneldo LOZA MENCÍA, Ioannis KATAKIS, Sang-Hyeun PARK, Johannes FÜRNKRANZ: *On the Combination of Two Decompositive Multi-Label Classification Methods*. In: *Proceedings of the ECML PKDD 2009 Workshop on Preference Learning (PL-09, Bled, Slovenia)*, eds. Eyke HÜLLERMEIER, Johannes FÜRNKRANZ, pp. 114–129, 2009. <http://www.ke.tu-darmstadt.de/events/PL-09/09-Tsoumakas.pdf>. (p. 8, 153)

## Technical Reports

Wouter DUIVESTELJN, Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ, Arno J. KNOBBE: *Multi-label LeGo – Enhancing Multi-label Classifiers with Local Patterns*. Tech. Rep. TUD-KE-2012-02, Knowledge Engineering Group, Technische Universität Darmstadt, 2012. <http://www.ke.tu-darmstadt.de/publications/reports/tud-ke-2012-02.pdf>. (p. 199)

Eneldo LOZA MENCÍA, Johannes FÜRNKRANZ: *Pairwise Learning of Multilabel Classifications with Perceptrons*. Tech. Rep. TUD-KE-2007-05, TU Darmstadt, Knowledge Engineering Group, 2007. <http://www.ke.tu-darmstadt.de/publications/reports/tud-ke-2007-05.pdf>. (p. 84, 105)

Eneldo LOZA MENCÍA, Sang-Hyeun PARK, Johannes FÜRNKRANZ: *Advances in Efficient Pairwise Multilabel Classification*. Tech. Rep. TUD-KE-2008-06, TU Darmstadt, Knowledge Engineering Group, 2008. <http://www.ke.tu-darmstadt.de/publications/reports/tud-ke-2008-06.pdf>. (p. 123)

## Diploma Thesis

Eneldo LOZA MENCÍA: *Paarweises Lernen von Multilabel-Klassifikationen mit dem Perzeptron-Algorithmus*. Diploma thesis, TU Darmstadt, Knowledge Engineering

---

Group, 2006. [http://www.ke.tu-darmstadt.de/lehre/arbeiten/diplom/2006/Loza\\_Eneldo.pdf](http://www.ke.tu-darmstadt.de/lehre/arbeiten/diplom/2006/Loza_Eneldo.pdf). (p. 7, 33, 45, 71, 77, 84, 209)

---

## Gracias – Danke – Thanks

Zuallererst möchte ich Johannes Fürnkranz danken, da er mir erst die Möglichkeit gegeben hat, zu forschen und zu promovieren, und der meinen Kollegen und mir dafür ein ausgezeichnetes Umfeld zur Verfügung stellt, in dem wir große kreative Freiheiten genießen. Ich danke auch dem Korreferenten meiner Dissertation, Eyke Hüllermeier, für die interessanten Treffen, Gespräche und Diskussionen, die einen wichtigen Beitrag zu dieser Arbeit geleistet haben. Ich danke natürlich auch Prof. Biemann, Prof. Strufe und Prof. Goesele, die drei weiteren Prüfer, für Ihr Interesse an meiner Dissertation und dafür, daß sie sich so kurzfristig und kurz vor der Sommerpause noch Zeit für meine Disputation genommen haben.

Thanks go also to Sanny, Greg, Ioannis, Wouter, Arno, and again to Juffi and Eyke for the cooperation in my scientific works, on which this dissertation mainly relies. Nicht direkt sichtbar, aber nicht minder wichtig, war die Hilfe von Gerard, der wahrscheinlich bis zur Disputation abgesehen von den Prüfern und mir als einziger die komplette Arbeit gelesen hat. Hoffentlich werden es bald mehr.

Natürlich danke ich meinen Kollegen und Freunden, die immer ein offenes Wort für jedwege Fragestellung hatten und mich teils von Anfang an auf diesem langen Weg begleitet haben. Danke Gunter, Lorenz, Freddy, Nik, Sanny, Dirk, Heiko, Kilian, Christian und ganz besonders natürlich Gabi, die eigentliche Geschäftsführerin des Fachgebiets, und ohne die auch bei mir viele Sachen nicht so gelaufen wären, wie sie sollten.

Für die kleine Feierrunde und das leibliche Wohl nach der Verteidigung sorgten meine Mutter, die eigens für dieses Ereignis aus Spanien gekommen ist, und Sergej. Beiden danke ich sehr dafür. Einen Dank auch an alle interessierten Zuhörer während der Disputation.

Por supuesto, no sólo a nivel existencial, le debo mucho agradecimiento a mis queridos padres Amparo y Miguel. Desde pequeño han confiado en mí y me han apoyado en mis haceres y mis decisiones. Su educación y confianza han contribuido en gran medida a que haya logrado llevar a cabo este trabajo. Gracias también a mi hermana Talía por todo el cariño y aprecio. Tengo mucha ilusión de ver y oír que tal le irá en su particular camino. Te deseo todo lo mejor y te daré todo mi apoyo.

Además le doy las gracias al resto de mi familia, especialmente a Feli y Laín, quienes me han apoyado mucho en y desde Alemania. Un agradecimiento y un beso muy grande también a mi novia Jenny. Gracias por acompañarme en esta aventura, gracias por alegrarme más la vida.

Le quiero dedicar este trabajo especialmente a mis abuelos, quienes desgraciadamente no han podido vivir la finalización de este proyecto pero que me han acompañado entusiasmadamente durante gran parte de él y seguramente estarían muy orgullosos de mí.



---

Ein nicht unbedeutender Zeitraum während der Promotion hat nichts mit purer wissenschaftlicher Arbeit zu tun, ist aber ebenso wichtig für ihren Erfolg. Die kleinen und großen Ablenkungen befreien den Geist von festgefahrenen Gedanken und machen Platz für frische Ideen. Für die langen Ablenkungen in und außerhalb der Universität sorgten insbesondere Freddy und Sebastian, mit denen ich unzählige abenteuerliche Reisen in nahen und fernen Ländern unternahm.

Aber auch die passenden Ablenkungen vor Ort waren wichtig für eine ausgewogene und einfallsreiche Promotion. Einen dankenden Gruß hierbei an die Frankfurter Truppe, die Darmstädter Fraktion, die Freunde aus dem spanisch-deutschen Erasmus, die Freunde in Aranda, Madrid, Spanien, Deutschland, Schweden und überall sonst auf der Welt.

---

## **Wissenschaftlicher Werdegang<sup>1</sup>**

- 09-1996 – 06-1998 Besuch der Oberstufe der Deutschen Schule Madrid,  
Allgemeine Hochschulreife und Selectividad
- 10-1998 – 03-2006 Studium der Informatik an der Technischen Universität Darmstadt,  
Abschluß zum Diplom-Informatiker
- 05-2006 – 04-2012 Wissenschaftlicher Mitarbeiter am  
Fachbereich Informatik der Technischen Universität Darmstadt

## **Erklärung zur Dissertation<sup>2</sup>**

Hiermit versichere ich, die vorliegende Dissertation selbständig nur mit den genannten Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 24. Juli 2012

Eneldo Loza Mencía

---

<sup>1</sup> Gemäß §20 Absatz 3 der Promotionsordnung der Technischen Universität Darmstadt.

<sup>2</sup> Gemäß §9 Absatz 1 der Promotionsordnung der Technischen Universität Darmstadt.